JOURNAL OF COMPUTER AND SYSTEM SCIENCES 46, 218–270 (1993)

Handle-Rewriting Hypergraph Grammars*

BRUNO COURCELLE

LaBRI, Université de Bordeaux 1, 351, Cours de la Libération, 33405 Talence, France

AND

JOOST ENGELFRIET AND GRZEGORZ ROZENBERG

Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands

Received June 15, 1990; revised August 27, 1991

We introduce the handle-rewriting hypergraph grammars (HH grammars), based on the replacement of handles, i.e., of subhypergraphs consisting of one hyperedge together with its incident vertices. This extends hyperedge replacement, where only the hyperedge is replaced. A HH grammar is separated (an S-HH grammar) if nonterminal handles do not overlap. The S-HH grammars are context-free, and the sets they generate can be characterized as the least solutions of certain systems of equations. They generate the same sets of graphs as the NLC-like vertex-rewriting C-edNCE graph grammars that are also context-free. © 1993 Academic Press, Inc.

INTRODUCTION

The concept of a context-free graph grammar is an attractive way of formalizing the notion of a recursively defined set of graphs, i.e., of a recursive graph property. Unfortunately, as opposed to the case of strings, there is not one type of context-free graph grammar, but there are many types, depending, e.g., on what kind of graph is generated or how the embedding works. Roughly speaking one may distinguish three types of context-free graph grammars, according to which part of a given graph is rewritten into another graph: a vertex, an edge, or a handle (i.e., an edge, together with its two incident vertices). In general one can say that handle-rewriting grammars are more powerful than vertex-rewriting grammars, which in their turn have more power than edge-rewriting grammars. To increase the power of edge-rewriting grammars, hyperedge-rewriting grammars have been introduced in [1, 27, 28] (and studied, e.g., in [8, 10, 18, 19, 25, 26, 33, 37]). Such grammars generate (directed) hypergraphs, of which the (hyper)edges are rewritten into hypergraphs. Still, these grammars are not as powerful as vertex-rewriting grammars, with the set of all complete graphs as a counterexample. In this paper we introduce the so-called separated handle-rewriting hypergraph grammar (S-HH

* Supported by ESPRIT BRWG 3299 "Computing by Graph Transformations."

0022-0000/93 $5.00

grammar). In such a grammar, a handle, i.e., a hyperedge, together with all its incident vertices, is rewritten into a hypergraph. The adjective "separated" means that the nonterminal handles (i.e., the handles that are rewritten) do not overlap, which means that they have no common incident vertices. S-HH grammars are a natural generalization of the hyperedge-rewriting grammars of [1, 27, 28]. We show, as our first main result, that the S-HH grammars have the same power as the largest known class of context-free vertex-rewriting grammars: the confluent edNCE grammars (or C-edNCE grammars) studied in, e.g., [3, 16, 32, 40]. C-edNCE grammars are a special case of the grammars of [38] and a generalization of the NLC grammars of [30, 31]; subclasses of C-edNCE grammars were considered in, e.g., [21–24].

A completely different type of graph grammar is the algebraic term-rewriting graph grammar [1, 5, 36]. The sentential forms of a term-rewriting grammar are terms rather than (hyper)graphs, i.e., they are expressions that denote graphs. A term-rewriting grammar is just a context-free (string) grammar (or a ground term-rewriting system) that generates a set of such expressions. To obtain the generated (hyper)graph language one just interprets these expressions as (hyper)graphs. We define three types of operations on hypergraphs or, rather, on "hypergraphs with ports," i.e., hypergraphs of which some of the vertices are labeled with integers. The operations are disjoint union, edge creation (using the integers to determine which edges are created), and port selection (that changes the integer labeling). We show, as our second main result, that, using the expressions built from these operators (together with two constants denoting the empty and the one-vertex hypergraph), term-rewriting grammars have the same power as S-HH grammars, and hence (in view of our first result), the same power as the vertex-rewriting C-edNCE grammars. Such an algebraic characterization did not yet exist for vertex-rewriting graph grammars (for hyperedge-rewriting grammars see [1]). One advantage of term-rewriting grammars is that, through their interpretation as the least fixed point of a system of equations, they correspond closely to the intuitive way in which recursive definitions of graph properties are usually given (whereas the ordinary graph grammars have a more operational, generative flavour). Another advantage is that they lead to results concerning monadic second-order logic, as in [10].

These three distinct descriptions of this class of graph languages (through S-HH grammars, C-edNCE grammars, and algebraic term-rewriting grammars), together with two other characterizations (through regular string/tree languages and through monadic second-order logic, see [16, 17]), demonstrate that it is a natural class of graph languages.

The separation property (which also holds for the sentential forms) guarantees that every S-HH grammar is confluent, i.e., that the order of applying productions does not influence the derived hypergraph. Confluence is a property that graph grammars should have to deserve the adjective context-free (see [6]). The edNCE grammars are not confluent, in general, which is the reason to restrict attention to confluent edNCE grammars (C-edNCE grammars). Since confluence is a dynamic,

operational property, it is rather difficult to work with. For this reason it is convenient to have an equivalent class of grammars (the S-HH grammars) for which no such restriction holds. To be more honest, we will first introduce arbitrary HH grammars, which are not confluent, and then impose confluence by requiring them to be separated. However, separation is a completely static, structural restriction.

This paper is divided into seven sections. The first section contains the necessary definitions of (directed, labeled) hypergraphs and graphs; the latter are conveniently defined as a special case of the former. In Section 2 HH grammars, and, in particular, S-HH grammars, are introduced, together with some of their elementary properties. After Section 2 the reader may either read Sections 3 and 4, in which the first main result is shown, or Sections 5 and 6, in which the second main result is shown. Section 3 contains the definition of C-edNCE grammars, and in Section 4 it is shown that S-HH grammars have the same graph generating power as the C-edNCE grammars. In Section 5 we define the three types of operations on hypergraphs with ports and we define the corresponding term-rewriting grammars, in the form of systems of equations (with least fixed point). Section 5 also contains a key lemma that is used in Section 6 to prove the equivalence of S-HH grammars and systems of equations. Finally, Section 7 discusses confluent NLC grammars (a subclass of the C-edNCE grammars investigated in [6]) and the hyperedge-rewriting grammars of [1, 28]. (A short version of this paper appeared in [12].)

We will use $\mathbb{N}$ to denote $\{0, 1, 2, ...\}$ and $\mathbb{N}_+$ to denote $\{1, 2, ...\}$. For $m, n \in \mathbb{N}$, $[m, n] = \{i \in \mathbb{N} \mid m \leqslant i \leqslant n\}$. For a finite set A, $\mathrm{card}(A)$ denotes its cardinality, and $\mathscr{P}(A)$ its powerset.

1. Hypergraphs and Graphs

Let A be an alphabet (of edge labels). A (directed, edge labeled) *hypergraph* over A is a tuple $H = (V, E)$, where V is the finite set of *vertices* and E is the finite set of *hyperedges*. Each hyperedge is a tuple $(a, v_1, ..., v_k)$ with $a \in A$, $k \geqslant 1$, and $v_i \in V$ for $i \in [1, k]$. We will often use "edge" instead of "hyperedge." $\mathrm{HG}(A)$ denotes the set of all hypergraphs over A.

Let $H = (V, E)$ be a hypergraph over A. Note that H may have multiple edges, but not with the same label. For an edge $e = (a, v_1, ..., v_k)$ in E, we write $\mathrm{vert}(e) = (v_1, ..., v_k)$, $\mathrm{vert}(e, i) = v_i$, and $\mathrm{vset}(e) = \{v_1, ..., v_k\}$. Each v_i is said to be a vertex of e, or a vertex incident with e. The number i is said to be a *tentacle* of e. Note that the v_i are not necessarily distinct. Note also that we do not allow $k = 0$. The integer $k \geqslant 1$ is called the *rank* of e, denoted $\mathrm{rank}(e)$, and a is called the label of e, denoted $\mathrm{lab}(e)$.

As usual, we will add a subscript H to indicate that we deal with the hypergraph H; thus, V_H stands for V, E_H for E, and, e.g., vert_H for vert.

Two hypergraphs H and K are *disjoint* if V_H and V_K are disjoint. Two hypergraphs H and K are *isomorphic* if they differ only with respect to the identity of

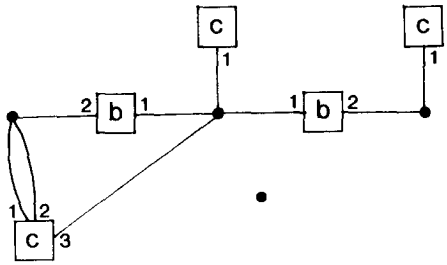

FIG. 1. A hypergraph.

their vertices; i.e., formally, if there is a bijection f: $V_H \to V_K$ such that $E_K = \{f(e) \mid e \in E_H\}$, where, for $e = (a, v_1, ..., v_k)$, $f(e) = (a, f(v_1), ..., f(v_k))$. Such a mapping is called an isomorphism.

EXAMPLE. A hypergraph (V, E) over $A = \{b, c\}$ is shown in Fig. 1. The set of vertices is $V = \{v_1, v_2, v_3, v_4\}$, where we have enumerated the vertices from left to right, and the set of edges is $E = \{(c, v_1, v_1, v_2), (b, v_2, v_1), (c, v_2), (b, v_2, v_4), (c, v_4)\}$. Note that v_3 is an isolated vertex.

In general, a vertex is drawn as a dot, and a hyperedge $(a, v_1, ..., v_k)$ is drawn as a square box with label a inside, together with a line labeled i (a tentacle) between the box and the dot representing v_i, for every $i \in [1, k]$. An edge (a, v_1, v_2) may also be drawn as an ordinary directed edge from v_1 to v_2, with label a, and an edge (a, v_1) may also be drawn as vertex label a of v_1; thus, Fig. 2 shows the same hypergraph as Fig. 1. These last conventions allow us to draw directed labeled graphs, defined next, in the usual way.

Let A be an alphabet (of edge and vertex labels). A (directed, edge labeled, vertex labeled) *graph* over A is a hypergraph $H = (V, E)$ over A such that (1) $\text{rank}(e) = 1$ or $\text{rank}(e) = 2$ for every $e \in E$, and (2) for every $v \in V$ there is exactly one hyperedge $e \in E$ such that $e = (a, v)$ for some $a \in A$. The hyperedges of rank 2 are the edges of the graph, whereas the hyperedges of rank 1 are the vertex labels. Thus, for graphs, the word "edge" will always refer to the hyperedges of rank 2. For $i = 1, 2$ we write

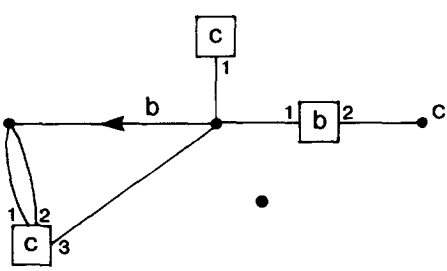

FIG. 2. The same hypergraph.

$E(i) = \{e \in E \mid \text{rank}(e) = i\}$. If $e = (a, v) \in E(1)$, then the label a of e is also called the label of v, denoted $\text{lab}(v)$; thus, $a = \text{lab}(e) = \text{lab}(v)$. If $e = (a, u, v) \in E(2)$, we say, as usual, that e leads from u to v. Note that loops, i.e., edges (a, u, u), are allowed. $\text{GR}(A)$ denotes the set of all graphs over A. Thus, $\text{GR}(A) \subseteq \text{HG}(A)$.

For any notion of isomorphism between objects, we use $[x]$ to denote the set of objects isomorphic to object x. For a set X of objects, $[X]$ denotes $\bigcup \{[x] \mid x \in X\}$, i.e., the set of all objects that are isomorphic to some object in X. Sometimes, x is called a concrete object and $[x]$, an abstract object. In particular, for a hypergraph H, $[H]$ is an abstract hypergraph. $[\text{HG}(A)]$ denotes the set of all abstract hypergraphs over A.

2. Handle Hypergraph Grammars

In context-free-like (hyper)graph grammars, a production is often of the form $X \to (H, \text{emb})$, where X is a nonterminal, H is a (hyper)graph, and emb is an embedding relation. The nonterminal X is meant to label a vertex or an edge, depending on whether the grammar rewrites vertices or edges. A rewriting step according to such a production consists of removing the vertex or edge labeled X from the given (hyper)graph, substituting H in its place, and connecting H to the remainder of the graph in a way specified by the embedding relation emb. A convenient way of treating this process formally is to add emb to H, to consider the pair (H, emb) as a new type of object, say, "hypergraph with embedding," and to view rewriting as a substitution operation on such objects (cf. [6]). Intuitively, these objects are quite natural: they are hypergraphs ready to be embedded in an environment. In this section we define a new type of graph grammar in this way: the handle hypergraph grammar (abbreviated HH grammar). It is a rather straightforward generalization of the context-free hypergraph grammar (or hyperedge replacement grammar) of [1, 26, 27, 28].

The sentential forms of a HH grammar are arbitrary hypergraphs, of which the edges are labeled by nonterminal and terminal symbols. However, as opposed to the context-free hypergraph grammar, the grammar is not edge rewriting but handle rewriting. This means that not only the (nonterminal) edge is removed, but also its incident vertices: a "handle" is an edge together with its incident vertices, cf. [15, 34] (do not confuse it with the notion of handle in parsing theory). Handle rewriting in hypergraphs may be viewed as a generalization of both (hyper)edge rewriting (in case the incident vertices are re-established) and vertex rewriting (in case the hyperedge is of rank 1), and is related to the handle rewriting of [35]. In HH grammars, the embedding is similar to that of NLC-like vertex rewriting grammars (such as, the edNCE grammars discussed in the next section), in the sense that edges in the environment may be duplicated or deleted. The embedding is also similar to, and may be viewed as a "multi-valued" version of, the one used in the context-free hypergraph grammars: each incident vertex of the rewritten edge is replaced by several vertices (gluing points), rather than just one. These gluing

points will be called "ports." We start by defining "hypergraphs with embedding," as discussed above. The embedding relation consists of a specification of the ports.

Let A be an alphabet. A *hypergraph with ports* over A is a pair (H, port) such that H is in $\text{HG}(A)$ and port is a finite subset of $\mathbb{N}_+ \times V_H$. $\text{HG}^P(A)$ denotes the set of all hypergraphs with ports over A. If $(i, v) \in \text{port}$, we say that v is an i-port (or just a port) and that i is a port number of v. Since port is a relation, we can employ the usual terminology for relations. Thus, $\text{port}(i)$ denotes $\{v \in V_H \mid (i, v) \in \text{port}\}$: the set of all i-ports, and $\text{port}^{-1}(v)$ denotes $\{i \in \mathbb{N}_+ \mid (i, v) \in \text{port}\}$: the set of all port numbers of a given vertex v. If $H = (V, E)$, we also write (V, E, port) for (H, port). If $\text{port} = \varnothing$, we identify (H, port) with H. Thus, every hypergraph is a hypergraph with ports. Two hypergraphs with ports (H, port_H) and (K, port_K) are isomorphic if there is an isomorphism f from H to K such that $\text{port}_K = \{(i, f(v)) \mid (i, v) \in \text{port}_H\}$.

EXAMPLE. Figure 3 shows a hypergraph with ports. To indicate the ports, each vertex v is labeled with the elements of $\text{port}^{-1}(v)$. To avoid confusion with a vertex label (a, v) we never use integers in A.

Intuitively, suppose that, in a hypergraph K, the hypergraph with ports (H, port_H) is substituted for a handle consisting of an edge e of K together with all its incident vertices. The substitution consists of (1) removing e and all its incident vertices from K, together with all edges incident with those vertices, (2) adding H to the remainder, and (3) embedding H in this remainder. The embedding is controlled by the ports in port_H, as follows. If an edge e' different from e was incident with the ith vertex of e, then for each vertex w of H that has port number i a "copy" of e' is created that will be incident with w. More precisely, each edge $(a, v_1, ..., v_k) \neq e$ of K is replaced by all edges $(a, w_1, ..., w_k)$ such that for $j \in [1, k]$: if $v_j \notin \text{vset}_K(e)$ then $w_j = v_j$, and if $v_j \in \text{vset}_K(e)$ then there exists $i \in [1, \text{rank}(e)]$ such that $v_j = \text{vert}_K(e, i)$ and $(i, w_j) \in \text{port}_H$.

Thus, the vertices in $\text{port}_H(i)$ are the gluing points that replace $\text{vert}_K(e, i)$. If $\text{port}_H(i)$ contains more than one vertex, then, intuitively, $\text{vert}_K(e, i)$ is duplicated, and if $\text{port}_H(i)$ is empty, $\text{vert}_K(e, i)$ is deleted.

In the above we have explained how to substitute a hypergraph with ports into a hypergraph (without ports). This idea is consistent with the fact that a hyper-

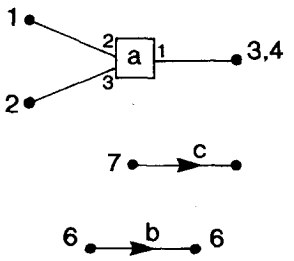

FIG. 3. A hypergraph with ports (H, port_H).

graph with ports formalizes the notion of a hypergraph ready to be substituted. To have a more uniform theory of substitution (see [6]) it is convenient to define the substitution of hypergraphs with ports into hypergraphs that also have ports (and hence one has to explain what the ports of the new object will be). For the reader who is interested in Sections 2–4 only, it suffices to understand this definition in the case that $\text{port}_K = \varnothing$ (in which case also $\text{port} = \varnothing$); the full definition is needed in Sections 5–7.

DEFINITION 2.1. Let (K, port_K) and (H, port_H) be two hypergraphs with ports in $\text{HG}^p(A)$, such that K and H are disjoint, and let e be an edge of K. Then the *substitution* of (H, port_H) for e in (K, port_K), denoted $(K, \text{port}_K)[(H, \text{port}_H)/e]$, is the hypergraph with ports (V, E, port) in $\text{HG}^p(A)$, defined as follows:

$$V = (V_K - \text{vset}(e)) \cup V_H.$$

For every vertex $v \in V_K$, define $\text{rep}(v) \subseteq V$ by

$$\text{rep}(v) = \begin{cases} \{v\} & \text{if} \quad v \notin \text{vset}(e) \\ \bigcup \{\text{port}_H(i) \mid v = \text{vert}(e, i), i \in [1, \text{rank}(e)]\} & \text{otherwise.} \end{cases}$$

(Intuitively, $\text{rep}(v)$ is the set of vertices by which v is replaced in V.)

$$E = E_H \cup \{(a, w_1, ..., w_k) \mid \exists e' \in E_K - \{e\}: e' = (a, v_1, ..., v_k),$$
$$w_i \in \text{rep}(v_i) \text{ for every } i \in [1, k]\}.$$
$$\text{port} = \{(i, w) \mid \exists v \in V_K: (i, v) \in \text{port}_K \text{ and } w \in \text{rep}(v)\}.$$

In other words, for every i, $\text{port}(i) = \bigcup \{\text{rep}(v) \mid v \in \text{port}_K(i)\}$.

Although the form of the above definitions of E and port does not reflect the "locality" of substitution as well as the one of V, it is short and hence convenient in proofs.

As an example, Fig. 4 shows a hypergraph (K, port_K) with ports, and Fig. 5 shows (an isomorphic copy of) the hypergraph $(K, \text{port}_K)[(H, \text{port}_H)/e]$, where (H, port_H) is the hypergraph with ports from Fig. 3 and e is the edge of K with label X.

DEFINITION 2.2. A *handle hypergraph grammar* (abbreviated HH grammar) is a tuple $G = (N, A, P, X_{\text{in}})$, where N is the nonterminal alphabet, A is the terminal alphabet (disjoint with N), P is the finite set of productions of the form $X \to (H, \text{port})$ with $X \in N$ and $(H, \text{port}) \in \text{HG}^p(N \cup A)$, and $X_{\text{in}} \in N$ is the initial nonterminal.

For a hypergraph $K \in \text{HG}(N \cup A)$ we say that $e \in E_K$ is a *nonterminal edge* if $\text{lab}_K(e) \in N$, and that it is a terminal edge otherwise. Moreover, we say that $v \in V_K$ is a *nonterminal vertex* if it is incident with a nonterminal edge, and a terminal

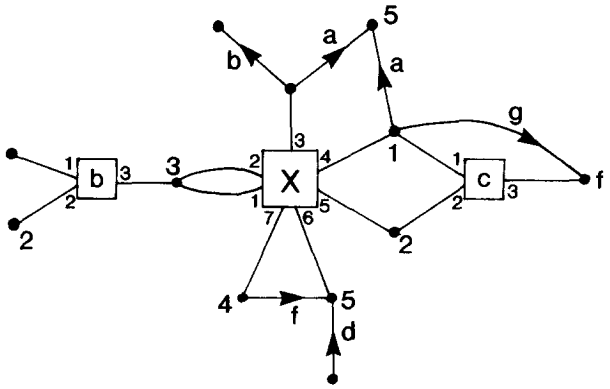

FIG. 4. Another hypergraph with ports (K, port_K).

vertex if it is not incident with any nonterminal edge. Two objects $p_i = X_i \to (H_i, \text{port}_i)$, $i = 1, 2$, where X_i is in N and (H_i, port_i) is a hypergraph with ports, are called isomorphic if $X_1 = X_2$ and (H_1, port_1), (H_2, port_2) are isomorphic. The elements of $[P]$ will also be called productions of G. For a production $p = X \to (H, \text{port})$ we use $\text{rhs}(p)$ to denote H.

The process of rewriting in the HH grammar G is defined through the notion of substitution in a standard language theoretic way, as follows: Let K and K' be hypergraphs over $N \cup A$, let $e \in E_K$, and let $p = X \to (H, \text{port})$ be a production of G (i.e., an element of $[P]$). Then we write $K \underset{(e,p)}{\Longrightarrow} K'$ if $\text{lab}_K(e) = X$ and $K' = K[(H, \text{port})/e]$. Note that K and H must be disjoint. A derivation $K_0 \underset{(e_1,p_1)}{\Longrightarrow} K_1 \underset{(e_2,p_2)}{\Longrightarrow} \cdots \underset{(e_n,p_n)}{\Longrightarrow} K_n$, $n \geq 0$, is *disjoint* if the hypergraphs in the set $\{K_0\} \cup \{\text{rhs}(p_i) \mid i \in [1, n]\}$ are mutually disjoint. We will restrict ourselves to disjoint derivations (without loss of generality, and without always mentioning it).

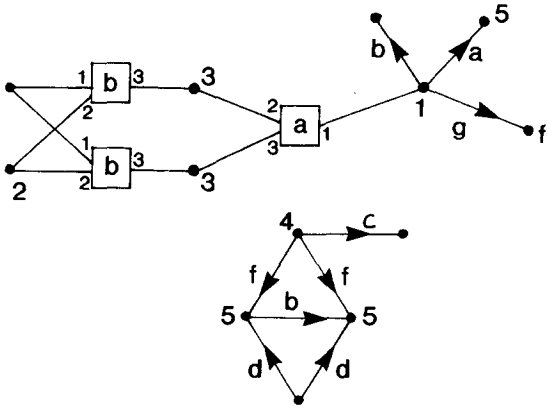

FIG. 5. The hypergraph with ports $(K, \text{port}_K)[(H, \text{port}_H)/e]$.

Thus, we write $K \overset{*}{\Rightarrow} K'$ if there is a disjoint derivation as above, with $K_0 = K$ and $K_n = K'$. Let $\mathrm{se}(X_{\mathrm{in}})$ denote a hypergraph with a single vertex v and a single edge (X_{in}, v). A *sentential form* of G is a hypergraph $K \in \mathrm{HG}(N \cup A)$ such that $\mathrm{se}(X_{\mathrm{in}}) \overset{*}{\Rightarrow} K$. The *hypergraph language generated by* G is $\mathrm{L}(G) = \{[K] \mid K \in \mathrm{HG}(A), \mathrm{se}(X_{\mathrm{in}}) \overset{*}{\Rightarrow} K\}$. Thus, sentential forms of G are concrete hypergraphs, but the language generated by G contains abstract hypergraphs, i.e., $\mathrm{L}(G) \subseteq [\mathrm{HG}(A)]$. Note that, since in the first step of each derivation $\mathrm{se}(X_{\mathrm{in}})$ is replaced in one stroke (because it is a handle), it is not essential that we have defined $\mathrm{se}(X_{\mathrm{in}})$ with a single vertex only.

Remark 2.3. Let $G = (N, A, P, X_{\mathrm{in}})$ be an HH grammar. In the (disjoint) derivations of G we have allowed the application of arbitrary productions in $[P]$. However, it should be clear that we do not need the full power of $[P]$, as long as there are sufficiently many isomorphic copies of productions in P. In fact, if $P' \subseteq [P]$ has the following property:

> for every $p \in P$ and every finite set U there exists $p' \in P' \cap [p]$ such that $V_{\mathrm{rhs}}(p')$ and U are disjoint,

then we may restrict attention to (disjoint) derivations in which productions from P' only are applied. We will call this *P'-restricted derivations*. In particular, disregarding derivations that are not P'-restricted does not change $\mathrm{L}(G)$.

In a sentential form K of an HH grammar G, the terminal vertices of K and all (terminal) edges between them cannot be altered any more by G. The other elements of K may be altered by G. An edge (either terminal or nonterminal) may be altered, even duplicated or deleted, if it is adjacent to a nonterminal edge (when this nonterminal edge is rewritten). A nonterminal vertex or edge is removed when a production is applied to the corresponding handle. In the example just before Definition 2.2, where a production $X \to (H, \mathrm{port}_H)$ is applied to nonterminal edge e of K (disregarding port_K), the b-labeled edge of rank 3 is duplicated (and so are the d- and f-labeled edges of rank 2), the c-labeled edge of rank 3 is deleted, and the two a-labeled edges of rank 2 are merged.

EXAMPLES. We give several examples of handle hypergraph grammars. All these grammars will in fact generate graphs. Figure 6 shows the three productions of HH grammar $G_1 = (N, A, P, X_{\mathrm{in}})$ with $N = \{X_1, X_2\}$, $X_{\mathrm{in}} = X_1$, and $A = \{a, b, c, q\}$. Figure 7 shows a derivation of a graph in $\mathrm{L}(G_1)$. It is not difficult to see that $\mathrm{L}(G_1)$ is the set of all "ladders" of the form shown in Fig. 7 (with an arbitrary number of "squares" rather than 3). Note that port(1) and port(2) are both singletons, in each of the productions for X_2. Thus, there is no duplication or deletion of nonterminal vertices (and their incident edges). In fact, G_1 is also a context-free hypergraph grammar (of [1, 27, 28]). Note, finally, that the first production is superfluous (i.e., X_2 may as well be taken as the initial nonterminal).

The productions of an HH grammar G_2 generating all complete directed graphs (without loops) are shown in Fig. 8. We have dropped the unique terminal label.

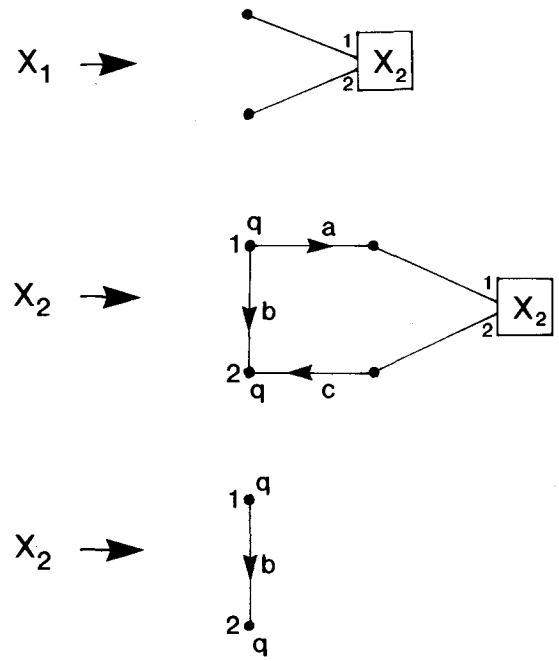

FIG. 6. The HH grammar G_1, generating all "ladders."

Application of the first production to a nonterminal hyperedge (X, v) results in the duplication of vertex v and all its incident edges. Figure 9 shows a derivation of the complete graph with four vertices. This language cannot be generated by a context-free hypergraph grammar (see Propositions 3.17 and 4.17 of [1]; see also [26, 27]).

By taking a unique terminal label and by viewing an undirected edge as two directed edges in opposite directions, we can generate undirected, unlabeled graphs. In pictures we will drop the terminal label, and we will draw an undirected edge in the usual way. Co-graphs are undirected, unlabeled graphs defined recursively as follows (see [4]): (1) a graph consisting of one vertex is a co-graph; (2) if H_1 and H_2 are (disjoint) co-graphs, then so is their (disjoint) union $H_1 + H_2$; (3) if H_1 and H_2 are (disjoint) co-graphs, then so is the graph $H_1 \times H_2$, obtained from $H_1 + H_2$ by adding all edges that connect a vertex of H_1 and a vertex of H_2. An HH grammar G_3 generating all co-graphs is shown in Fig. 10, where production π_i corresponds to case (i) of the above definition. A derivation in G_3 of the square is shown in Fig. 11.

HH grammar G_4 in Fig. 12 generates all "dotted" trees, i.e., all undirected binary trees, of which the vertices are labeled q, together with one additional vertex labeled s that is connected by an edge to every vertex of the binary tree. The initial nonterminal X_{in} generates the s-labeled vertex, together with the root of the tree. The second production adds two new leaves as the sons of an old leaf. G_4 is also a context-free hypergraph grammar.

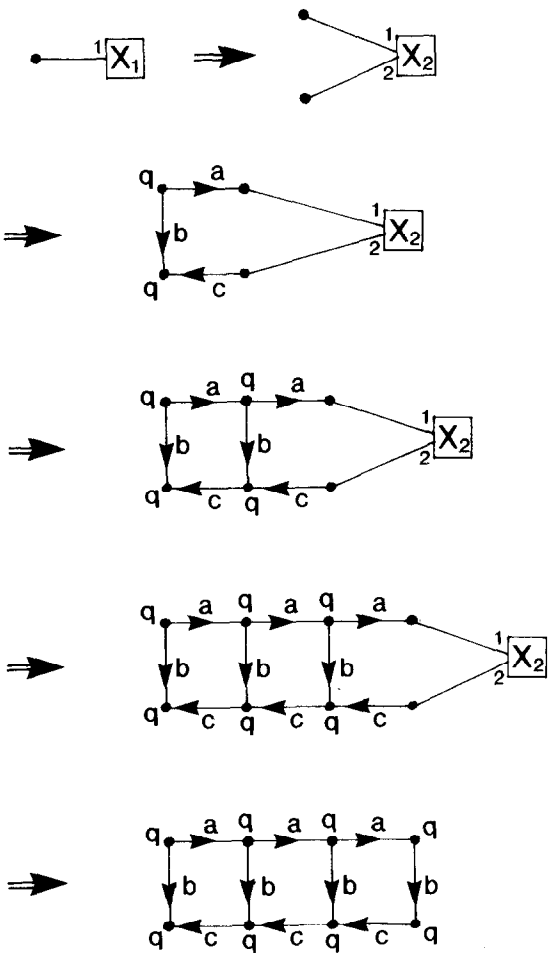

FIG. 7. A derivation of a "ladder" in G_1.

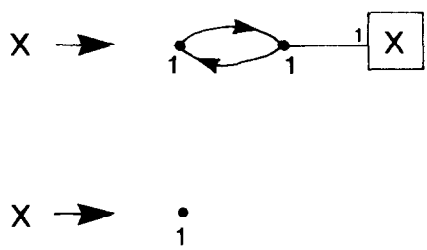

FIG. 8. The HH grammar G_2, generating all complete graphs.

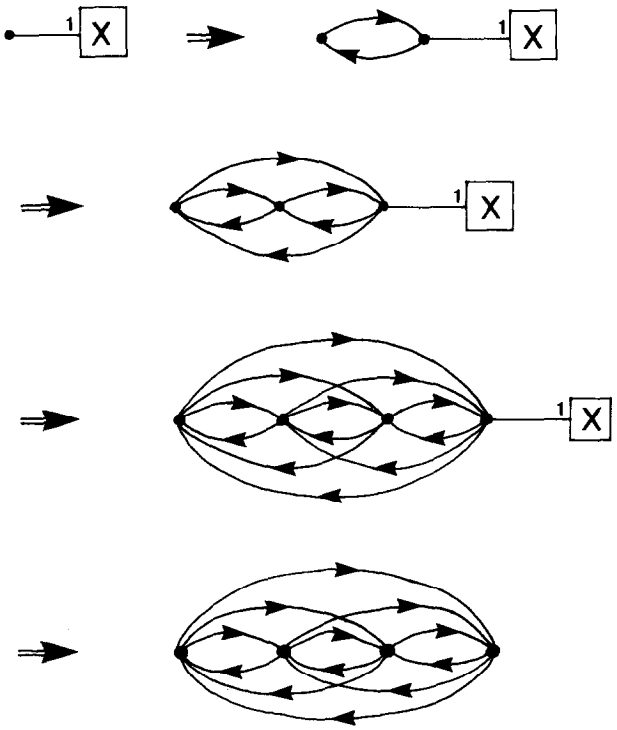

FIG. 9. A derivation of a complete graph in G_2.

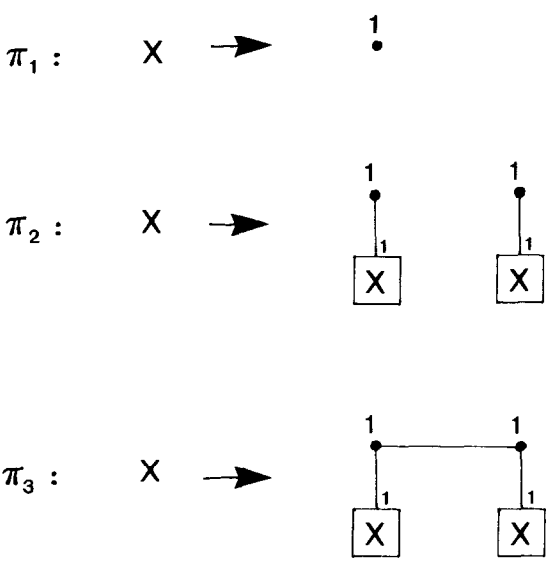

FIG. 10. The HH grammar G_3, generating all co-graphs.

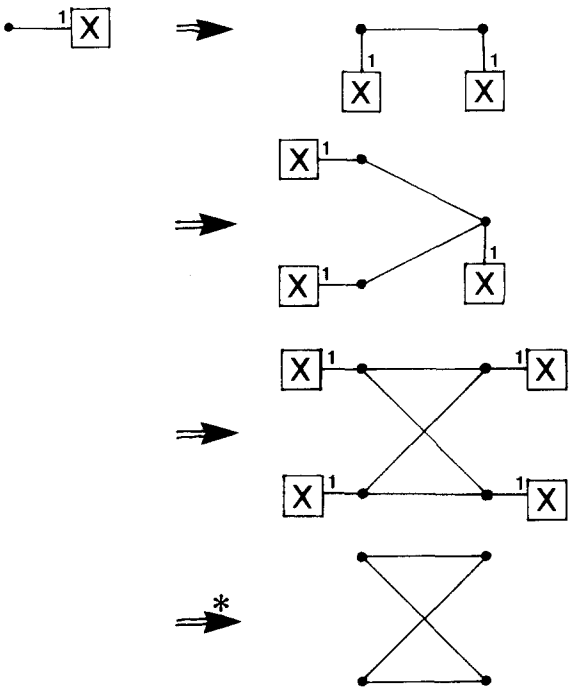

FIG. 11. A derivation of a co-graph in G_3 and G_8.

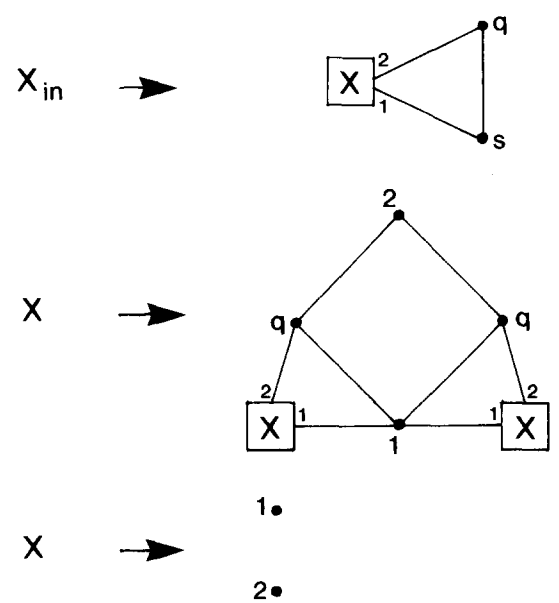

FIG. 12. The HH grammar G_4, generating all "dotted" trees.

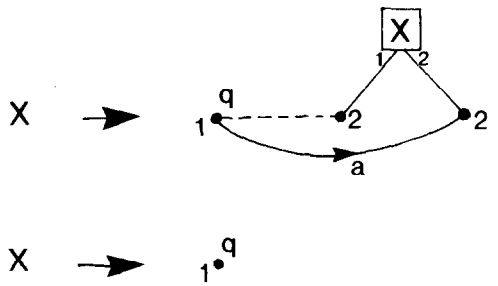

FIG. 13. The HH grammar G_5, generating all (uni-directional) edge complements of chains.

HH grammar G_5 in Fig. 13 generates chains of vertices, where a vertex has an edge from every vertex to its left and to every vertex to its right, except its immediate neighbours. The dotted lines should be disregarded; they represent "missing" edges. A derivation in G_5 is shown in Fig. 14. Thus, when disregarding the direction of the edges (and the labels), G_5 generates all edge complements of chains.

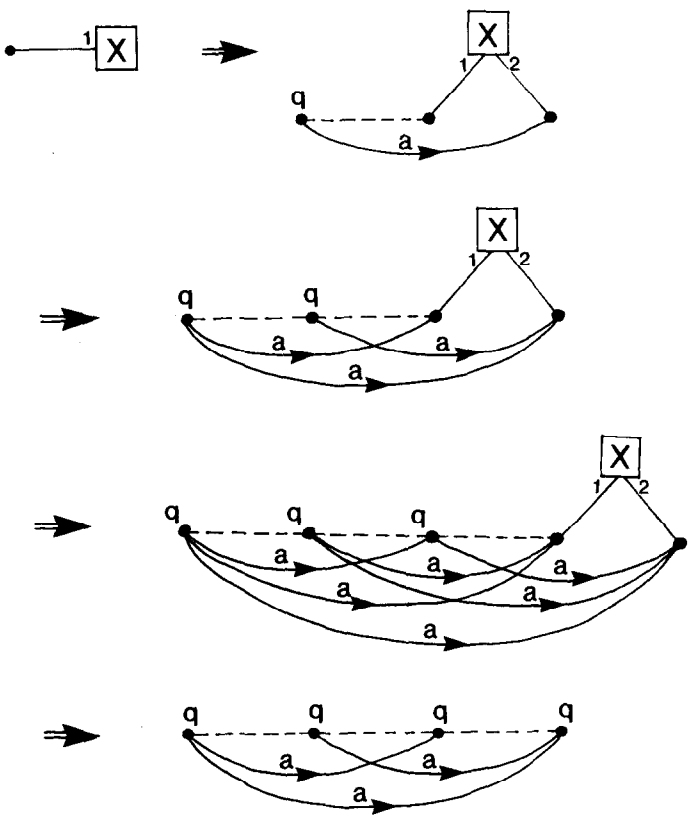

FIG. 14. A derivation of a (uni-directional) edge complement of a chain in G_5.

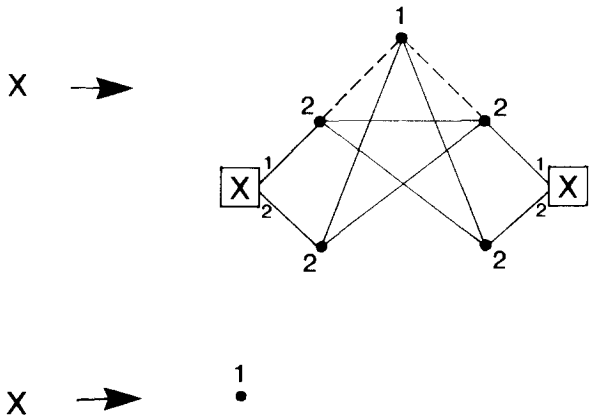

FIG. 15. The HH grammar G_6, generating all edge complements of binary trees.

HH grammar G_6 in Fig. 15 is similar to G_5: it generates all edge complements of (undirected, unlabeled) binary trees. For a nonterminal edge e in a sentential form, vert$(e, 1)$ is a leaf of the binary tree, whereas vert$(e, 2)$ is an auxiliary vertex that is connected with all vertices of the tree except vert$(e, 1)$. Again, the dotted lines should be disregarded; they represent the edges of the binary tree. Showing a derivation in this grammar would not be very helpful, due to the large number of edges.

Although rewriting in HH grammars has the context-free property that a production with left-hand side X is always applicable to an edge labeled X, it is not context-free in the sense that rewriting of one nonterminal edge may influence the remaining nonterminal edges (they may be duplicated or deleted). One, rather drastic but quite natural, way to avoid this is to require that no two nonterminal edges share incident vertices (this is similar to the so-called boundary restriction for the vertex-rewriting NLC graph grammars: no two nonterminal vertices share an incident edge [39]). We say that a hypergraph (with ports) over $N \cup A$ is *separated* if no vertex is incident with two distinct nonterminal edges. In other words, in a separated hypergraph all nonterminal handles are disjoint: $\text{vset}(e) \cap \text{vset}(e') = \varnothing$ for every two distinct nonterminal edges e and e'.

DEFINITION 2.4. An HH grammar is *separated* (abbreviated S-HH grammar) if all right-hand sides of its productions are separated.

The only example grammar that is not separated is G_4. It is easy to see that all sentential forms of an S-HH grammar are separated too. Due to this property, the notion of a separated HH grammar is an example of the abstract notion of grammar discussed in Section 2 of [6]. Separated HH grammars form the topic of investigation of this paper.

Some types of graph grammar (such as the NLC grammar) are not context-free

in the sense that the result of applying two productions to distinct nonterminal edges/vertices of a sentential form may depend on the order in which these productions are applied. Grammars for which the application of productions is order independent are said to be *confluent* [6, 32] or to have the finite Church–Rosser property [3, 39, 40]. As argued in Theorem 2.11 of [6], confluent grammars allow the use of derivation trees. We now show that S-HH grammars are confluent.

LEMMA 2.5. *Let G be an* S-HH *grammar, and let K be a sentential form of G. If $K \underset{(e_1, p_1)}{\Longrightarrow} K_1 \underset{(e_2, p_2)}{\Longrightarrow} K_{12}$ and $K \underset{(e_2, p_2)}{\Longrightarrow} K_2 \underset{(e_1, p_1)}{\Longrightarrow} K_{21}$ are (disjoint) derivations with e_1, $e_2 \in E_K$, then $K_{12} = K_{21}$.*

Proof. Obviously, e_1 and e_2 are distinct nonterminal edges of K. Let $p_i = X_i \to (H_i, \text{port}_i)$, for $i = 1, 2$. H_1 and H_2 are disjoint, because the derivations are disjoint. Now, $K_{12} = K[(H_1, \text{port}_1)/e_1][(H_2, \text{port}_2)/e_2]$ and $K_{21} = K[(H_2, \text{port}_2)/e_2][(H_1, \text{port}_1)/e_1]$. Since K is separated, it can easily be checked that these (concrete) hypergraphs are equal. ∎

We now turn to two lemmas that provide a normal form for S-HH grammars. An edge f of a hypergraph over $N \cup A$ is a *parasite* if there is a nonterminal edge $e \neq f$ such that $\text{vset}(f) \subseteq \text{vset}(e)$. Note that if e is rewritten, the embedding process may establish edges (corresponding to f) that lie completely within the right-hand side of the applied production. We say that a hypergraph over $N \cup A$ is parasite-free if it has no parasites. An HH grammar is *parasite-free* if all right-hand sides of its productions are parasite-free (all example grammars, except G_4, are parasite-free). It is easy to see that the sentential forms of a parasite-free S-HH grammar are all parasite-free.

LEMMA 2.6. *For every* S-HH *grammar G there is a parasite-free* S-HH *grammar G' such that $L(G') = L(G)$.*

Proof. Let $G = (N, A, P, X_{\text{in}})$ be an S-HH grammar. We will construct $G' = (N', A, P', X'_{\text{in}})$. The (obvious) idea is to remove the (terminal!) parasites from the productions and sentential forms of G and to keep track of them in the labels of the nonterminal edges. When such a nonterminal edge is rewritten, the appropriate embedding edges can be added (statically) to the right-hand side of the production applied.

We first define the nonterminal alphabet N' of G'. Let m be the maximal rank of the edges in the right-hand sides of the productions of G. Now, $N' = \{(X, R) \mid X \in N,\ R \subseteq \{(a, i_1, i_2, ..., i_k) \mid a \in A,\ k \in [1, m],\ i_j \in [1, m] \text{ for all } j \in [1, k]\}\}$. Intuitively, if (X, R) labels an edge e, an element $(a, i_1, ..., i_k)$ of R represents the terminal edge $(a, \text{vert}(e, i_1), ..., \text{vert}(e, i_k))$.

We now define a function g: $\text{HG}(N \cup A) \to \text{HG}(N' \cup A)$ that replaces the parasites by the corresponding information. For every separated $H \in \text{HG}(N \cup A)$ of

which the edges have rank $\leqslant m$, $g(H)$ is the parasite-free separated hypergraph that is obtained from H in the following steps:

— if e is a nonterminal edge with label X, then relabel it by (X, R), where $R = \{(a, i_1, ..., i_k) \mid a \in A,\ (a, \mathrm{vert}_H(e, i_1), ..., \mathrm{vert}_H(e, i_k)) \in E_H\}$,

— remove all parasites, i.e., all terminal edges of the form $(a, \mathrm{vert}(e, i_1), ..., \mathrm{vert}(e, i_k))$ for some nonterminal edge e.

Next, we extend g to hypergraphs with ports, for every R (with $(X, R) \in N'$ for some $X \in N$). Let $(H, \mathrm{port}) \in \mathrm{HG}^p(N \cup A)$ be a hypergraph with ports, such that H is separated and has edges of rank at most m. Then $g_R(H, \mathrm{port})$ is defined to be $(g(H_R), \mathrm{port})$, where H_R is obtained from H by adding all (terminal) edges $(a, v_1, ..., v_k)$ such that there exists $(a, i_1, ..., i_k) \in R$ with $(i_j, v_j) \in \mathrm{port}$ for all $j \in [1, k]$.

Finally, we complete the construction of G' by defining $X'_{\mathrm{in}} = (X_{\mathrm{in}}, \varnothing)$ and $P' = \{(X, R) \to g_R(H, \mathrm{port}) \mid (X, R) \in N',$ and $X \to (H, \mathrm{port})$ is in $P\}$.

It can be shown that $g(K[(H, \mathrm{port})/e]) = g(K)[g_R(H, \mathrm{port})/e']$, where e' is the edge corresponding to e, i.e., $\mathrm{lab}_{g(K)}(e') = (\mathrm{lab}_K(e), R)$ and $\mathrm{vert}_{g(K)}(e') = \mathrm{vert}_K(e)$. From this it follows (by induction on the length of the derivations) that, for every $K' \in \mathrm{HG}(N' \cup A)$,

$$\mathrm{se}(X'_{\mathrm{in}}) \overset{*}{\Rightarrow} K' \text{ in } G' \text{ iff}$$

$$\exists K \in \mathrm{HG}(N \cup A)\colon \mathrm{se}(X_{\mathrm{in}}) \overset{*}{\Rightarrow} K \text{ in } G \text{ and } K' = g(K).$$

For the only–if direction note that $[P'] = \{(X, R) \to g_R(H, \mathrm{port}) \mid (X, R) \in N',$ and $X \to (H, \mathrm{port})$ is in $[P]\}$. Since g is the identity on hypergraphs in $\mathrm{HG}(A)$, this proves that $\mathrm{L}(G') = \mathrm{L}(G)$. ∎

A nonterminal edge e of a hypergraph is *loop-free* if $\mathrm{vert}(e, 1), ..., \mathrm{vert}(e, k)$ are all distinct (where $k = \mathrm{rank}(e)$). We say that a hypergraph is loop-free if all its nonterminal edges are loop-free. An HH grammar is *loop-free* if all right-hand sides of its productions are loop-free (all example grammars are loop-free). Obviously, the sentential forms of a loop-free S-HH grammar are all loop-free.

LEMMA 2.7. *For every* S-HH *grammar* G *there is a loop-free and parasite-free* S-HH *grammar* G' *such that* $L(G') = L(G)$.

Proof. Informally, the obvious idea is as follows. If $\mathrm{vert}(e, i) = \mathrm{vert}(e, j)$ for $i \neq j$, then we split this vertex into two and duplicate all incident edges. Let $G = (N, A, P, X_{\mathrm{in}})$ be an S-HH grammar. By Lemma 2.6 we may assume that G is parasite-free. We construct $G' = (N, A, P', X_{\mathrm{in}})$, where P' is obtained with the help of the translation $g\colon \mathrm{HG}^p(N \cup A) \to \mathrm{HG}^p(N \cup A)$ that splits vertices and duplicates edges, formally defined as follows. For every separated $(H, \mathrm{port}) \in \mathrm{HG}^p(N \cup A)$, $g(H, \mathrm{port})$ is the loop-free separated hypergraph with ports (V, E, port') such that

$$V = \{v \in V_H \mid v \text{ is terminal}\} \cup \{(e, i) \mid e \in E_H \text{ is nonterminal}, i \in [1, \mathrm{rank}(e)]\},$$

$$E = \{(\mathrm{lab}_H(e), (e, 1), ..., (e, k)) \mid e \in E_H \text{ is nonterminal}, k = \mathrm{rank}(e)\}$$
$$\cup \{(a, v_1, ..., v_k) \mid (a, \mathrm{old}(v_1), ..., \mathrm{old}(v_k)) \in E_H \text{ is terminal}\},$$

where $\text{old}(v) = v$ if v is a terminal vertex of H, and $\text{old}(e, i) = \text{vert}_H(e, i)$,

$$\text{port}' = \{(i, v) \mid (i, \text{old}(v)) \in \text{port}\}.$$

Note that g preserves parasite-freeness. Now we set $P' = \{X \to g(H, \text{port}) \mid X \to (H, \text{port}) \text{ is in } P\}$. Thus, G' is loop-free and parasite-free.

It can be shown, for separated hypergraphs K and (H, port), and a nonterminal edge e of K, that $g(K[(H, \text{port})/e]) = g(K)[g(H, \text{port})/e']$, where $e' = (\text{lab}_H(e), (e, 1), \ldots, (e, k))$ with $k = \text{rank}(e)$. From this follows, as in the proof of Lemma 2.6, that $L(G') = L(G)$. The only difference is that in this case we have to restrict the derivations of G' to $g([P])$-restricted derivations, where $g([P]) = \{X \to g(H, \text{port}) \mid X \to (H, \text{port}) \text{ is in } [P]\}$. It is easy to check that this is allowed, according to Remark 2.3. ▮

3. The edNCE Graph Grammars

One of our main results is that separated HH grammars have the same graph generating power as confluent edNCE graph grammars. The edNCE grammars are vertex rewriting grammars that have been studied both as a restriction of the grammars of [38] (in [2, 3, 32, 40]) and as a generalization of the NLC grammars of [30, 31] (in, e.g., [16, 21, 22, 23, 24, 25]). By restricting edNCE grammars to be confluent one obtains a class of grammars that is the largest known class of context-free graph grammars, where "largest" refers to the corresponding class of graph languages (and "context-free" is meant in the sense of [6], which excludes, e.g., the context-free graph grammars of [38]).

In this section we define the class of confluent edNCE grammars in the way sketched in the introduction of Section 2. NCE stands for "neighbourhood controlled embedding," e for "edge labeled graphs" (it being understood that they are vertex labeled), and d for "directed graphs." We start with the appropriate definition of "graphs with embedding," i.e., a graph ready to be substituted.

Let A be an alphabet. A *graph with* (neighbourhood controlled) *embedding* over A is a pair (H, emb) with $H \in \text{GR}(A)$ and $\text{emb} \subseteq V_H \times A \times A \times A \times \{\text{in}, \text{out}\}$. $\text{GR}^e(A)$ denotes the set of all such graphs with embedding. If $H = (V, E)$, then (H, emb) will also be written (V, E, emb). If $\text{emb} = \varnothing$, we identify (H, emb) with H. Thus, $\text{GR}(A) \subseteq \text{GR}^e(A)$. Two graphs with embedding (H, emb_H) and (K, emb_K) are isomorphic if there is an isomorphism f from H to K such that $\text{emb}_K = \{(f(u), a, b, q, d) \mid (u, a, b, q, d) \in \text{emb}_H\}$.

Intuitively, a tuple $(u, a, b, q, \text{out}) \in \text{emb}$ means that if there was an a-labeled edge from the vertex v for which (H, emb) is substituted to a vertex w with label q, then the embedding process will establish a b-labeled edge from u to w. And similarly for in instead of out (where "in" refers to incoming edges of v and "out" to outgoing edges of v). Thus, as opposed to HH grammars, the embedding looks at edge labels and changes them; it also looks at the direction of the edge and at the label of the neighbouring vertex.

We now define this formally. The reader who is not interested in Section 7 may take both emb_K and emb below to be empty. Recall that, for a graph (V, E), $E(2)$ denotes its set of edges and $E(1)$ its vertex labeling.

DEFINITION 3.1. Let (K, emb_K) and (H, emb_H) be two graphs with embedding, in $\mathrm{GR}^e(A)$, such that K and H are disjoint, and let v be a vertex of K. Then the *substitution* of (H, emb_H) for v in (K, emb_K), denoted $(K, \mathrm{emb}_K)[(H, \mathrm{emb}_H)/v]$, is the graph with embedding (V, E, emb) in $\mathrm{GR}^e(A)$, defined as follows:

$$V = (V_K - \{v\}) \cup V_H,$$

$$E(1) = \{(a, w) \in E_K(1) \mid a \in A, w \neq v\} \cup E_H(1),$$

$$E(2) = \{(a, u, w) \in E_K(2) \mid a \in A, u \neq v, w \neq v\} \cup E_H(2)$$
$$\cup \{(b, u, w) \mid \exists a \in A: (a, v, w) \in E_K(2), (u, a, b, \mathrm{lab}_K(w), \mathrm{out}) \in \mathrm{emb}_H\}$$
$$\cup \{(b, w, u) \mid \exists a \in A: (a, w, v) \in E_K(2), (u, a, b, \mathrm{lab}_K(w), in) \in \mathrm{emb}_H\},$$

$$\mathrm{emb} = \{(u, a, b, q, d) \in \mathrm{emb}_K \mid u \neq v\}$$
$$\cup \{(u, a, c, q, d) \mid \exists b \in A: (v, a, b, q, d) \in \mathrm{emb}_K, (u, b, c, q, d) \in \mathrm{emb}_H\}.$$

An elementary property of substitution of graphs with embedding is that of *associativity*, which means that the following lemma holds.

LEMMA 3.2. *Let M, K, H be three mutually disjoint graphs with embedding, in $\mathrm{GR}^e(A)$. Let w be a vertex of M and v a vertex of K. Then $M[K/w][H/v] = M[K[H/v]/w]$.*

The proof is a straightforward but lengthy verification from the definitions and will not be given (a similar proof is given in Lemma 5.3 of [6]). We note that substitution of separated hypergraphs with ports is associative too.

The definition of edNCE grammar is now analogous to that of HH grammar.

DEFINITION 3.3. An *edNCE* grammar is a tuple $G = (N, A, P, X_{\mathrm{in}})$, where N is the nonterminal alphabet, A is the terminal alphabet (disjoint with N), P is the finite set of productions of the form $X \to (H, \mathrm{emb})$ with $X \in N$ and $(H, \mathrm{emb}) \in \mathrm{GR}^e(N \cup A)$, and $X_{\mathrm{in}} \in N$ is the initial nonterminal.

For a graph $K \in \mathrm{GR}(N \cup A)$ we say that $v \in V_K$ is a *nonterminal vertex* if $\mathrm{lab}_K(v) \in N$, and that it is a terminal vertex otherwise. As in the case of HH grammars we denote by $[P]$ the class of productions that are isomorphic to a production in P. For a production $p = X \to (H, \mathrm{emb})$ we use $\mathrm{rhs}(p)$ to denote H.

We now define the process of rewriting in an edNCE grammar through the notion of substitution, in the usual way. Let K and K' be graphs over $N \cup A$, let $v \in V_K$, and let $p = X \to (H, \mathrm{emb})$ be in $[P]$. Then we write $K \underset{(v,p)}{\Longrightarrow} K'$ if $\mathrm{lab}_K(v) = X$ and $K' = K[(H, \mathrm{emb})/v]$. As in the case of HH grammars we restrict ourselves to disjoint derivations. Let $\mathrm{sv}(X_{\mathrm{in}})$ denote a graph with a single vertex,

labeled X_{in}, and no edges. A graph $K \in GR(N \cup A)$ such that $sv(X_{in}) \overset{*}{\Rightarrow} K$ is a *sentential form* of G. The *graph language generated by* G is $L(G) = \{[K] \mid K \in GR(A), sv(X_{in}) \overset{*}{\Rightarrow} K\}$. Thus, $L(G) \subseteq [GR(A)]$.

EXAMPLES. Figure 16 shows the productions of an edNCE grammar $G_7 = (N, A, P, X_1)$, with $N = \{X_1, X_2, \alpha, \beta\}$, that generates the same set of 'ladders" as the HH grammar G_1. For each production $X \to (H, \text{emb})$ the picture shows both H and emb; emb is represented by the dotted labeled lines, in a rather obvious way. The second production has $H = (V, E)$ with $V = \{x, y, z\}$ and $E = \{(b, x, y), (\alpha, x, z), (\beta, z, y), (q, x), (q, y), (X_2, z)\}$, and $\text{emb} = \{(x, \alpha, a, q, \text{in}), (y, \beta, c, q, \text{out})\}$. A derivation in G_7 of a ladder with two "squares" is shown in Fig. 17. The first production is superfluous: X_2 might as well have been the initial nonterminal.

Figure 18 shows the productions of an edNCE grammar G_8 generating all co-graphs (i.e., the same language as HH grammar G_3). Figure 11 is also a derivation in G_8 (disregarding labels).

Figure 19 shows edNCE grammar G_9 that generates the same graphs as HH grammar G_5 (the uni-directional edge complements of chains). Just as G_5, G_9 generates the vertices one by one from left to right. The unique nonterminal vertex is connected by an α-labeled edge to all terminal vertices except the last, and by a β-labeled edge to all terminal vertices.

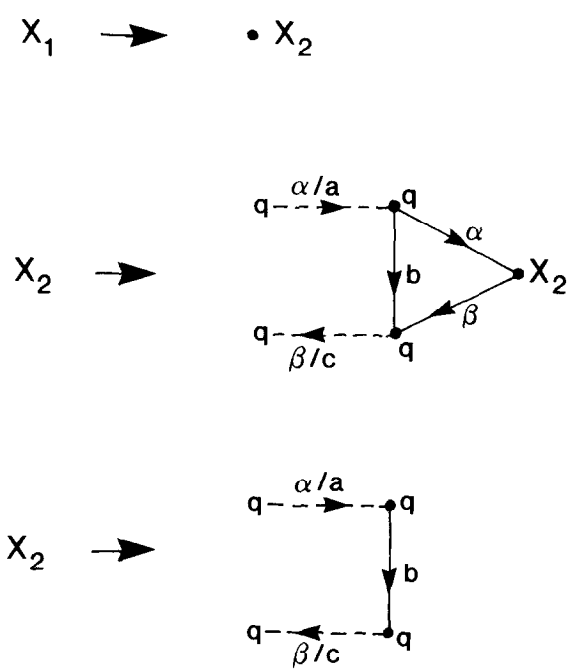

FIG. 16. The edNCE grammar G_7, generating all "ladders."

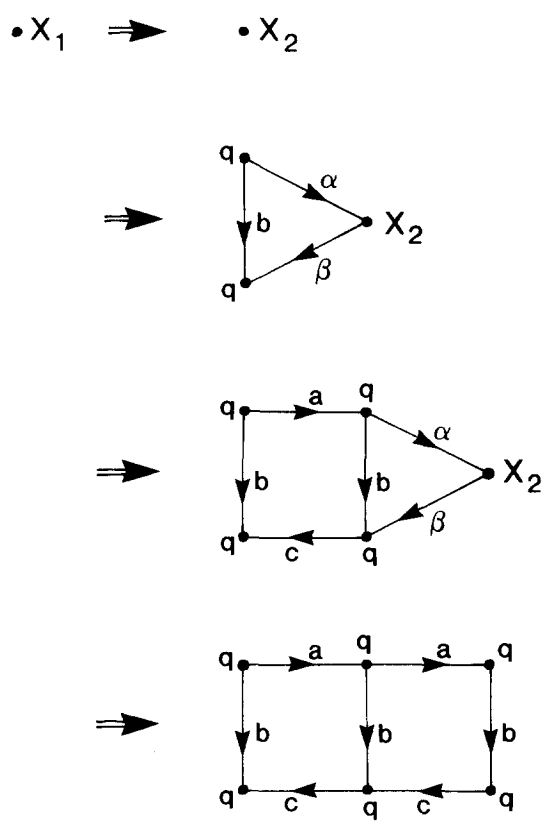

FIG. 17. A derivation of a "ladder" in G_7.

As an important class of edNCE grammars we consider confluent edNCE grammars, i.e., those grammars in which the application of productions is order independent. It is well known that NLC grammars (a subclass of edNCE grammars) are not always confluent. Confluent NLC grammars were studied in [6], and confluent edNCE grammars in [3, 16, 32, 40] (called fCR DNELC grammars in [3]). Since our notion of substitution is associative (Lemma 3.2), confluent edNCE grammars (or C-edNCE grammars) might also be called context-free edNCE grammars, in the sense of [6]. S-HH grammars are context-free in the same sense.

DEFINITION 3.4. An edNCE grammar $G = (N, A, P, X_{in})$ is *confluent* (abbreviated C-edNCE) if the following holds for every sentential form K of G: if $K \underset{(v_1, p_1)}{\Longrightarrow} K_1 \underset{(v_2, p_2)}{\Longrightarrow} K_{12}$ and $K \underset{(v_2, p_2)}{\Longrightarrow} K_2 \underset{(v_1, p_1)}{\Longrightarrow} K_{21}$ are (disjoint) derivations with $v_1, v_2 \in V_K$ and $v_1 \neq v_2$, then $K_{12} = K_{21}$.

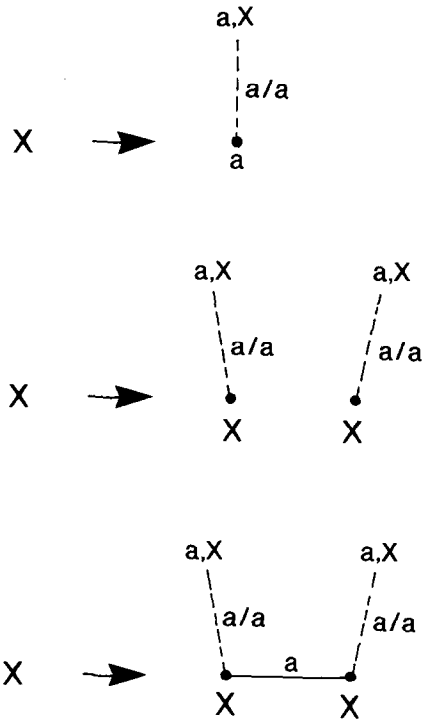

FIG. 18. The edNCE grammar G_8, generating all co-graphs.

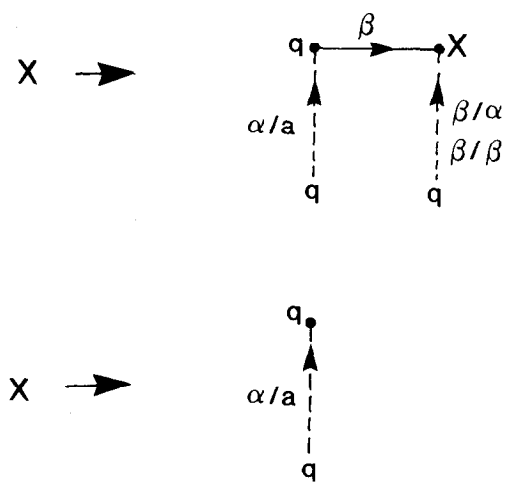

FIG. 19. The edNCE grammar G_9, generating all (uni-directional) edge complements of chains.

By C-edNCE we denote the class of graph languages generated by C-edNCE grammars. It is decidable whether an edNCE grammar is confluent (Corollary 3.25 of [32]), but we shall not need this property. G_7, G_8, and G_9 are all confluent (in G_7 and G_9 each sentential form contains at most one nonterminal vertex, and in G_8 the embeddings contain all possible tuples).

It is well known that "boundary" graph grammars are confluent (see Lemma 2.3 or [39]). An edNCE grammar is *boundary* (abbreviated B-edNCE) if in no right-hand side of a production there is an edge between two distinct nonterminal vertices. G_7 and G_9 are trivially boundary, but G_8 is not. It is easy to show that B-edNCE grammars are confluent. Thus, B-edNCE $\subseteq$ C-edNCE.

An important property of confluent grammars is that every derivation is equivalent to a "leftmost" derivation. The only way we know to simulate a C-edNCE grammar G by an S-HH grammar G' is such that G' simulates the leftmost derivations of G: in this way G' has control over the dynamic relabeling of edges in G. To define leftmost derivations in a precise way, it is convenient to put a linear order on the nonterminal vertices of sentential forms (and the nonterminal vertices of the right-hand side of a production), see [6]. In order not to burden our terminology and notation, we will do this in a rather informal way that should be understandable to the reader.

An *ordered graph* (possibly with embedding) is a graph, together with a (linear) order of its nonterminal vertices (if it has such vertices). If K and (H, emb_H) are ordered and v is a nonterminal vertex of K, then the order of $K[(H, \mathrm{emb}_H)/v]$ is obtained, informally, by substituting the order of H for v in the order of K. Formally, if the nonterminal vertices of K are ordered as $(v_1, ..., v_k)$ with $v = v_i$, and those of H are ordered as $(w_1, ..., w_h)$, then the order of $K[(H, \mathrm{emb}_H)/v]$ is $(v_1, ..., v_{i-1}, w_1, ..., w_h, v_{i+1}, ..., v_k)$.

The concept of an *ordered edNCE grammar* is obtained by considering ordered graphs throughout the definition of edNCE grammar. It should be clear that the order has no influence on the language generated, i.e., an ordered edNCE grammar generates the same language as the edNCE grammar that is obtained from it by disregarding the orders. However, for an ordered edNCE grammar G there is a natural notion of leftmost derivation. A derivation step $K \underset{(v,p)}{\Longrightarrow} K'$ in G is *leftmost* if v is the first nonterminal vertex in the order of K. A derivation is leftmost if all its steps are. We write $K \overset{*}{\underset{\mathrm{lm}}{\Longrightarrow}} K'$ if there is a leftmost (disjoint) derivation from K to K'. The *graph language leftmost generated by* G is $\mathrm{L}_{\mathrm{lm}}(G) = \{[K] \mid K \in \mathrm{GR}(A), \mathrm{sv}(X_{\mathrm{in}}) \overset{*}{\underset{\mathrm{lm}}{\Longrightarrow}} K\}$. By L-edNCE we denote the class of all graph languages leftmost generated by ordered edNCE grammars.

PROPOSITION 3.5. C-edNCE $\subseteq$ L-edNCE. *In particular, if G is an ordered* C-edNCE *grammar, then* $L_{\mathrm{lm}}(G) = L(G)$.

This proposition can be proved in a standard way (cf., e.g., [6]). In fact, since the grammar is confluent, the order of two consecutive independent applications of productions can be interchanged without changing the result of the derivation. It

turns out that in our simulation of the leftmost derivations of a C-edNCE grammar G by an S-HH grammar G' we will not any more make use of the confluence of G. Thus, that proof shows that even L-edNCE $\subseteq$ S-HH. Hence, as a corollary we obtain that C-edNCE = L-edNCE (cf. also [16]). In the case of NLC grammars, L-NLC contains languages not even in NLC (see [29]). Note that NLC grammars can be viewed as a special case of edNCE grammars (cf. Definition 7.1); hence L-NLC $\subseteq$ L-edNCE.

4. Simulation of HH Grammars by edNCE Grammars and Vice Versa

In this section we show that S-HH grammars and C-edNCE grammars have the same graph generating power (and, in fact, the same as L-edNCE grammars too). An HH grammar $G = (N, A, P, X_{in})$ is *graph generating* if $L(G) \subseteq [GR(A)]$. By GRL we denote the class of all graph languages. Thus, S-HH $\cap$ GRL is the class of all languages generated by graph generating S-HH grammars. We will show that S-HH $\cap$ GRL = C-edNCE = L-edNCE. We start with the simulation of graph generating S-HH grammars by C-edNCE grammars.

Lemma 4.1. S-HH $\cap$ GRL $\subseteq$ C-edNCE.

Proof. Let $G = (N, A, P, X_{in})$ be a graph generating S-HH grammar. By Lemma 2.7 we may assume that G is loop-free and parasite-free (in fact, in the construction that follows, it suffices to take G parasite-free). Moreover, since G is graph generating, we may assume that in each right-hand side of a production (and hence in each sentential form) every terminal hyperedge has rank $\leqslant 2$ (just delete the terminal hyperedges of greater rank). Note finally that, since G is parasite-free, nonterminal vertices of a sentential form K of G are not incident with hyperedges of rank 1; also, each terminal vertex of K is incident with exactly one hyperedge of rank 1; and the same holds for right-hand sides of productions.

We simulate G by an edNCE grammar $G' = (N', A, P', X_{in})$, with $N \subseteq N'$. The idea is to simulate each nonterminal hyperedge e of G by a nonterminal vertex v of G', with the same label, and to simulate a terminal hyperedge of rank 2 that is incident with vert(e, i), by an edge that is incident with v, with i encoded in the label of that edge. Nonterminal vertices of G are removed, and terminal vertices of G are carried over to G' (with the same label, i.e., the same incident terminal hyperedge of rank 1). This simulation can be defined by a direct translation of the sentential forms and productions of G into those of G'. See Fig. 20 for an example.

Let m be the maximal rank of all nonterminal hyperedges in the right-hand sides of the productions of G. The nonterminal alphabet N' of G' is defined by $N' = N \cup M$, where $M = (A \times [0, m] \times [0, m]) - (A \times \{0\} \times \{0\})$. The elements of N will be used to label vertices, and those of M to label edges. To simplify the construction that follows we will identify $\langle a, 0, 0\rangle$ with a, for every $a \in A$. Thus, $M \cup A = A \times [0, m] \times [0, m]$ and $N' \cup A = N \cup (A \times [0, m] \times [0, m])$. For

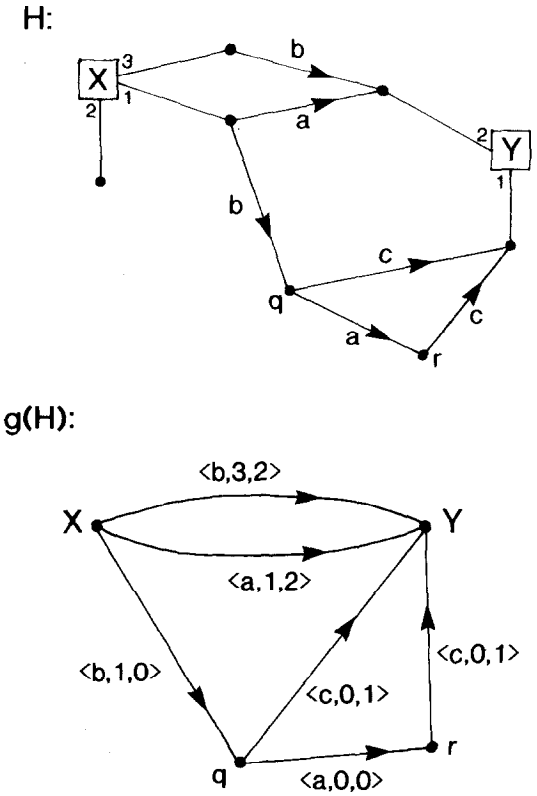

FIG. 20. From hypergraphs to graphs.

the same reason we define $\mathrm{vert}(v, 0) = v$ for every terminal vertex v in a hypergraph H from $\mathrm{HG}(N \cup A)$. Intuitively, for $\langle a, i, j\rangle \in M \cup A$, i refers to the ith vertex of a nonterminal hyperedge of H in case $i \geqslant 1$, or to a terminal vertex of H in case $i = 0$ (and similarly for j).

We now define the translation g: $\mathrm{HG}^p(N \cup A) \to \mathrm{GR}^e(N' \cup A)$. For a (loop-free and) parasite-free separated hypergraph with ports $(H, \mathrm{port}) \in \mathrm{HG}^p(N \cup A)$ (of which the nonterminal hyperedges have rank $\leqslant m$, and each terminal vertex is incident with exactly one hyperedge of rank 1), $g(H, \mathrm{port})$ is the graph with embedding $(K, \mathrm{emb}) \in \mathrm{GR}^e(N' \cup A)$ defined as

$$V_K = \{e \in E_H \mid e \text{ is nonterminal}\} \cup \{v \in V_H \mid v \text{ is terminal}\},$$

$$E_K(1) = \{(X, e) \mid X = \mathrm{lab}_H(e)\} \cup \{(a, v) \in E_H(1) \mid v \text{ is terminal}\},$$

$$E_K(2) = \{(\langle a, i, j\rangle, x, y) \mid (a, \mathrm{vert}_H(x, i), \mathrm{vert}_H(y, j)) \in E_H\},$$

$$\mathrm{emb} = \{(x, \langle a, i, j\rangle, \langle a, k, j\rangle, q, \mathrm{out}) \mid (i, \mathrm{vert}_H(x, k)) \in \mathrm{port}\}$$
$$\cup \{(x, \langle a, j, i\rangle, \langle a, j, k\rangle, q, \mathrm{in}) \mid (i, \mathrm{vert}_H(x, k)) \in \mathrm{port}\},$$

where the appropriate specifications should be added to the definitions of E_K and emb (in particular, $q \in N \cup A$).

We conclude the definition of G' by putting $P' = \{X \to g(H, \text{port}) \mid X \to (H, \text{port}) \text{ is in } P\}$.

It can now be shown that, for appropriate hypergraphs K and (H, port), and a nonterminal edge $e \in E_K$, $g(K[(H, \text{port})/e]) = g(K)[g(H, \text{port})/e]$. In other words, g is a homomorphism with respect to substitution (for such hypergraphs). From this it follows by induction that, for every $K' \in \text{GR}(N' \cup A)$,

$$\text{sv}(X_{\text{in}}) \overset{*}{\Rightarrow} K' \qquad \text{in } G'$$

iff

$$\exists K \in \text{HG}(N \cup A): \text{se}(X_{\text{in}}) \overset{*}{\Rightarrow} K \qquad \text{in } G, \qquad K' = g(K),$$

provided we consider $g([P])$-restricted derivations in G', where $g([P]) = \{X \to g(H, \text{port}) \mid X \to (H, \text{port}) \text{ is in } [P]\}$. Since g is the identity on graphs in $\text{GR}(A)$, it now follows that $\text{L}(G') = \text{L}(G)$. Confluence of G' easily follows from the confluence of G (Lemma 2.5), using again the facts that g is a homomorphism with respect to substitution, and that, in G', we may consider $g([P])$-restricted derivations only. ▮

As an example, consider the graph generating S-HH grammar G_5. The construction in the proof of Lemma 4.1 produces the C-edNCE grammar G_9 (apart from useless tuples in the embedding relations), taking $\alpha = \langle a, 0, 1\rangle$ and $\beta = \langle a, 0, 2\rangle$. Similarly, G_7 corresponds to G_1, with $\alpha = \langle a, 0, 1\rangle$ and $\beta = \langle c, 2, 0\rangle$; see also Figs. 7 and 17.

To show that C-edNCE $\subseteq$ S-HH and even that L-edNCE $\subseteq$ S-HH (as observed before), we need a simple closure property for S-HH.

LEMMA 4.2. *If* $L \in$ S-HH, $L \subseteq [\text{GR}(A)]$, *and* $B \subseteq A$, *then* $L \cap [\text{GR}(B)] \in$ S-HH.

Proof. Let $G = (N, A, P, X_{\text{in}})$ be an S-HH grammar that generates L. We will call a terminal hyperedge e (of rank 1 or 2) a non-B-edge if $\text{lab}(e) \notin B$. It is not correct to remove from P all productions of which the right-hand side contains a non-B-edge. In fact, the non-B-edge may be deleted later in the derivation. Thus, we first construct an S-HH grammar G' with $\text{L}(G') = \text{L}(G)$, such that the label of every nonterminal hyperedge of a sentential form of G' contains the information which of its vertices are incident with a non-B-edge.

Let m be the maximal rank of the nonterminal hyperedges in the right-hand sides of the productions of G. We define $G' = (N', A, P', X'_{\text{in}})$ where $N' = \{(X, T) \mid X \in N, T \subseteq [1, m]\}$, and $X'_{\text{in}} = (X_{\text{in}}, \varnothing)$. For every $T \subseteq [1, m]$ we define a mapping $g_T: \text{HG}^p(N \cup A) \to \text{HG}^p(N' \cup A)$ such that, for every $(H, \text{port}) \in \text{HG}^p(N \cup A)$, $g_T(H, \text{port})$ is obtained from (H, port) as follows: if e is a nonterminal hyperedge of H with label X, then relabel e by (X, T_e), where $T_e = \{i \in [1, m] \mid \text{vert}_H(e, i)$ is incident with a non-B-edge, or $\text{vert}_H(e, i)$ is a j-port for some $j \in T\}$.

We now define $P' = \{(X, T) \to g_T(H, \text{port}) \mid (X, T) \in N',\ X \to (H, \text{port}) \text{ is in } P\}$. It is easy to show that, for every $K' \in \text{HG}(N' \cup A)$,

$$\text{se}(X'_{\text{in}}) \overset{*}{\Rightarrow} K' \qquad \text{in } G'$$

iff

$$\exists K \in \text{HG}(N \cup A)\colon \text{se}(X_{\text{in}}) \overset{*}{\Rightarrow} K \qquad \text{in } G, \qquad K' = g_{\varnothing}(K).$$

Hence, an S-HH grammar G'' such that $\text{L}(G'') = \text{L}(G) \cap [\text{GR}(B)]$ is obtained from G' by removing from P' all productions $(X, T) \to g_T(H, \text{port})$ such that either there is a non-B-edge between terminal vertices of $g_T(H, \text{port})$, or there is a terminal i-port of $g_T(H, \text{port})$ for some $i \in T$. ∎

We now show that L-edNCE $\subseteq$ S-HH. To do this we also need leftmost derivations for S-HH grammars. An *ordered separated hypergraph* (possibly with ports) is a separated hypergraph together with an order of its nonterminal hyperedges. As for edNCE grammars, we can define leftmost derivations and $\text{L}_{\text{lm}}(G)$ for an (ordered) S-HH grammar G. Since every S-HH grammar is confluent (Lemma 2.5), it follows that $\text{L}_{\text{lm}}(G) = \text{L}(G)$, as for C-edNCE grammars (Proposition 3.5).

LEMMA 4.3. L-edNCE $\subseteq$ S-HH.

Proof. Let $G = (N, A, P, X_{\text{in}})$ be an ordered edNCE grammar. We may assume that in the right-hand sides of the productions (and in the sentential forms) of G no edge is labeled by a nonterminal: by an easy edge relabeling we can see to it that N is partitioned into N_{e} and N_{v}, where the elements of N_{e} label edges only, and those of N_{v} label vertices only; then we can apply Lemma 4.2, using the fact that $\text{L}_{\text{lm}}(G) = \text{L}_{\text{lm}}(G') \cap [\text{GR}(A)]$, where $G' = (N_{\text{v}}, A \cup N_{\text{e}}, P, X_{\text{in}})$.

We will simulate the leftmost derivations of G by the leftmost derivations of an S-HH grammar $G' = (N, A, P', X_{\text{in}})$. Note that G' has the same nonterminals as G. Again, the simulation of G by G' will be by a direct translation of the sentential forms and productions of G. Nonterminal vertices of G will be translated into nonterminal hyperedges of G' (with an appropriate number of incident vertices), and terminal vertices of G will be turned into terminal vertices of G' (with the same label). But the translation of the edges is quite complicated. The main problem is the simulation of the dynamically changing edge labels of G in G' (where edge labels cannot change). In G, an edge (a, u, v) between nonterminal vertices u and v can be changed into many differently labeled edges (a', u', v') between descendants u' and v' of u and v, respectively. As long as u' and v' are nonterminal, we will simulate these edge labels in G' by "tentacle names" (coded as integers) of the nonterminal hyperedges corresponding to u' and v'.

The set T of "tentacle names" is defined by $T = A \times (N \cup A) \times \{\text{in}, \text{out}\} \times A \times A$. Let m be the cardinality of T. We identify the elements of T with the integers in $[1, m]$ in a (arbitrarily) fixed way. This will enable us, e.g., to speak about $\text{vert}(e, t)$, where $t \in T$.

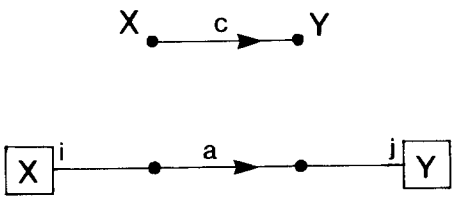

FIG. 21. From graphs to hypergraphs; $i = \langle c, Y, \text{out}, q, b\rangle$ and $j = \langle b, q, \text{in}, q', a\rangle$ for every guess of $\langle q, b\rangle$ and $\langle q', a\rangle$.

Let us now try to explain the idea behind the translation g: $\mathrm{GR}^e(N \cup A) \to \mathrm{HG}^p(N \cup A)$, where we assume all (hyper)graphs involved to be ordered. Let $H \in \mathrm{GR}(N \cup A)$. The nonterminal vertices of H are turned into nonterminal hyperedges of rank m in $g(H)$, with their order carried over, and, as observed before, the terminal vertices of H are turned into terminal vertices of $g(H)$, with the same label (i.e., hyperedge of rank 1). Each nonterminal hyperedge e of $g(H)$ gets m new (private) vertices that are incident with e. Now suppose that there is an edge (c, u, v) in H such that u and v are nonterminal vertices, and u is, say, to the left of v (in the order of the nonterminal vertices of H). Let X and Y be the labels of u and v, respectively. Intuitively, we would like to reverse the construction in the proof of Lemma 4.1 and translate this edge into a hyperedge (of rank 2) between two incident vertices of u and v in $g(H)$, i.e., a hyperedge $(a, \text{vert}(u, i), \text{vert}(v, j))$, see Fig. 21. In this way, we guess the label a of one of the edges (between terminal descendants of u and v) that the c-edge will finally turn into. In the tentacle names i and j we have to encode sufficient information for u and v to be able to simulate the leftmost derivations of u and v in G independent of each other in G'. Thus, we will take $i = \langle c, Y, \text{out}, q, b\rangle \in T$, where $\langle c, Y, \text{out}\rangle$ encodes the information needed by u about the original c-edge, and $\langle q, b\rangle$ is a guess concerning the label q of the terminal descendant u' of u to which the edge will be attached and the label b of the edge at that moment. Furthermore, we take $j = \langle b, q, \text{in}, q', a\rangle \in T$, where $\langle b, q, \text{in}\rangle$ is the information about the edge at the moment that v "takes over" the leftmost derivation from u', consistent with the guess of $\langle q, b\rangle$ in i, and $\langle q', a\rangle$ is a guess concerning the final edge (between terminal vertices), similar to the guess $\langle q, b\rangle$ in i (but note that a was already guessed to be its label; moreover, the guess of q' is in fact not needed, but added to simplify the formal construction). Thus, for every possible guess of a, q, b, and q', a terminal hyperedge (of rank 2) is added to $g(H)$, as described above.

This ends the informal discussion. We now continue with the formal definition of g. For every ordered graph with embedding $(H, \text{emb}_H) \in \mathrm{GR}^e(N \cup A)$, $g(H, \text{emb}_H)$ is the ordered separated hypergraph with ports $(K, \text{port}_K) \in \mathrm{HG}^p(N \cup A)$ defined as follows. First, $V_K = \{v \in V_H \mid v \text{ is terminal}\} \cup \{(v, i) \mid v \in V_H \text{ is nonterminal}, i \in T\}$. Second, $E_K = \mathrm{NE}_K \cup \mathrm{TE}_K(1) \cup \mathrm{TE}_K(2)$, where NE_K is the set of nonterminal hyperedges of K, and $\mathrm{TE}_K(1)$ and $\mathrm{TE}_K(2)$ are the sets of terminal hyperedges of K, of rank 1 and 2. $\mathrm{NE}_K = \{(\text{lab}_H(v), (v, 1), \ldots, (v, m)) \mid v \in V_H \text{ is nonterminal}\}$ and NE_K

inherits the order of the nonterminal vertices of H in the obvious way. $\mathrm{TE}_K(1) = \{(a, v) \in E_H(1) \mid a \in A\}$, and $\mathrm{TE}_K(2)$ is constructed as follows. Recall that $T = A \times (N \cup A) \times \{\mathrm{in}, \mathrm{out}\} \times A \times A$. Let (c, u, v) be an edge of H. We consider five cases:

(1) If u and v are terminal vertices of H, then (c, u, v) is in $\mathrm{TE}_K(2)$.

(2) If u and v are distinct nonterminal vertices of H, and u is to the left of v, then, for every $\langle q, b\rangle$, $\langle q', a\rangle \in A \times A$, $\mathrm{TE}_K(2)$ contains the hyperedge $(a, \mathrm{vert}_K(u, \langle c, \mathrm{lab}_H(v), \mathrm{out}, q, b\rangle), \mathrm{vert}_K(v, \langle b, q, \mathrm{in}, q', a\rangle))$.

(3) Similarly, if u and v are distinct nonterminal vertices of H, and u is to the right of v, then, for every $\langle q, b\rangle$, $\langle q', a\rangle \in A \times A$, $\mathrm{TE}_K(2)$ contains the hyperedge $(a, \mathrm{vert}_K(u, \langle b, q, \mathrm{out}, q', a\rangle), \mathrm{vert}_K(v, \langle c, \mathrm{lab}_H(u), \mathrm{in}, q, b\rangle))$.

(4) If u is a nonterminal vertex and v a terminal vertex of H, then $\mathrm{TE}_K(2)$ contains the hyperedge $(a, \mathrm{vert}_K(u, \langle c, \mathrm{lab}_H(v), \mathrm{out}, q', a\rangle), v)$, for every $\langle q', a\rangle \in A \times A$. (Intuitively, $\langle c, \mathrm{lab}_H(v), \mathrm{out}\rangle$ encodes the necessary information about the edge (c, u, v), and $\langle q', a\rangle$ is a guess concerning the final edge label a and the label q' of the terminal vertex to which it is attached, see Fig. 22.)

(5) Similarly, if u is a terminal vertex and v a nonterminal vertex of H, then, for every $\langle q', a\rangle \in A \times A$, $\mathrm{TE}_K(2)$ contains the hyperedge $(a, u, \mathrm{vert}_K(v, \langle c, \mathrm{lab}_H(u), \mathrm{in}, q', a\rangle))$.

This ends the construction of E_K. It remains to construct the set port_K, as follows: Let $(x, c, c', Y, d) \in \mathrm{emb}_H$. Thus, $x \in V_H$, $c, c' \in A$, $Y \in N \cup A$, and $d \in \{\mathrm{in}, \mathrm{out}\}$. We consider two cases:

(1) If x is a terminal vertex of H, then port_K contains $(\langle c, Y, d, \mathrm{lab}_H(x), c'\rangle, x)$. (Intuitively, here the guess $\langle \mathrm{lab}_H(x), c'\rangle$ is verified.)

(2) If x is a nonterminal vertex of H, then, for every $\langle q, b\rangle \in A \times A$, port_K contains $(\langle c, Y, d, q, b\rangle, \mathrm{vert}_K(x, \langle c', Y, d, q, b\rangle))$.

This ends the definition of the translation g. We complete the construction of G' by defining $P' = \{X \to g(H, \mathrm{emb}) \mid X \to (H, \mathrm{emb}) \text{ is in } P\}$.

As in the proof of Lemma 4.1 it can now be shown, for (ordered) graphs K and (H, emb), and a nonterminal vertex v of K, that $g(K[(H, \mathrm{emb})/v]) = g(K)[g(H, \mathrm{emb})/v]$, provided v is the first nonterminal vertex in the order of K. Thus, g is a

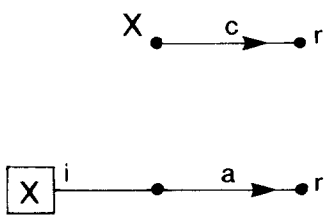

FIG. 22. From graphs to hypergraphs; $i = \langle c, r, \mathrm{out}, q', a\rangle$ for every guess of $\langle q', a\rangle$.

homomorphism with respect to "leftmost substitution." From this it follows that, for every (ordered) $K' \in \mathrm{HG}(N \cup A)$,

$$\mathrm{se}(X_{\mathrm{in}}) \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} K' \qquad \text{in } G'$$

iff

$$\exists(\text{ordered}) K \in \mathrm{GR}(N \cup A)\colon \mathrm{sv}(X_{\mathrm{in}}) \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} K \qquad \text{in } G, \qquad K' = g(K),$$

where only $g([P])$-restricted leftmost derivations in G' are considered. Since g is the identity on graphs in $\mathrm{GR}(A)$, it follows that $\mathrm{L}_{\mathrm{lm}}(G') = \mathrm{L}_{\mathrm{lm}}(G)$. Hence, since G' is confluent (Lemma 2.5), $\mathrm{L}(G') = \mathrm{L}_{\mathrm{lm}}(G)$. ▮

As an example, consider the C-edNCE grammar G_9 and take $N = \{X\}$ and $A = \{a, \alpha, \beta\}$. The construction in the proof of Lemma 4.3 produces the S-HH grammar G_5 (apart from useless tentacles of X), where tentacle 1 corresponds to tentacle name $\langle \alpha, q, \text{in}, q, a \rangle$ and tentacle 2 to $\langle \beta, q, \text{in}, q, a \rangle$ (note that $\langle q, a \rangle$ is the only useful guess). Similarly, G_1 corresponds to G_7, with tentacle 1 corresponding to $\langle \alpha, q, \text{in}, q, a \rangle$ and tentacle 2 to $\langle \beta, q, \text{out}, q, c \rangle$; see also Figs. 7 and 17.

Our first main result follows from Proposition 3.5 and Lemmas 4.1 and 4.3.

THEOREM 4.4. $\text{S-HH} \cap \mathrm{GRL} = \text{C-edNCE}$.

COROLLARY 4.5. $\text{C-edNCE} = \{\mathrm{L}(G) \mid G$ *is an* S-HH *grammar satisfying* $P_{\mathrm{gra}}\}$, *where* P_{gra} *means that for every production* $X \to (H, \text{port})$ *of* G

(1) *the rank of a terminal edge of* H *is at most* 2,

(2) *nonterminal vertices of* H *are not incident with terminal edges of rank* 1,

(3) *each terminal vertex of* H *is incident with exactly one terminal edge of rank* 1.

Proof. (If) It is easy to see that (1)–(3) also hold for sentential forms H of G.
(Only if) See the first paragraph of the proof of Lemma 4.1. ▮

Thus, S-HH grammars with property P_{gra} can be used instead of C-edNCE grammars. The advantage is that they are defined by easy structural properties, rather than by the dynamic property of confluence in the case of C-edNCE grammars.

We end this section by discussing the boundary case. As observed in Section 3, an edNCE grammar is boundary (B-edNCE) if in no right-hand side of a production there is an edge between two distinct nonterminal vertices (see, e.g., [21, 24]). Similarly, we now define an HH grammar to be *boundary* (abbreviated B-HH) if the following holds for every right-hand side H of a production: if e and e' are distinct nonterminal hyperedges of H, and $i \in [1, \mathrm{rank}(e)]$, $i' \in [1, \mathrm{rank}(e')]$, then there is no edge of H that is incident with both $\mathrm{vert}(e, i)$ and $\mathrm{vert}(e', i')$. Obviously,

every B-HH grammar is an S-HH grammar. Example grammars G_3, G_6, and G_8 are not boundary, while G_1, G_2, G_5, G_7, and G_9 are (trivially). It can easily be checked in all constructions involved in the proof of Theorem 4.4 that the boundary property is preserved. Thus, the following characterization of B-edNCE by HH grammars is obtained: B-edNCE = B-HH $\cap$ GRL. Note that although we have defined the notion of a separated HH grammar in analogy with a boundary edNCE grammar (cf. the remarks just before Definition 2.4), it has the same power as a C-edNCE grammar. It is shown in [24] that the set of edge complements of binary trees is not in B-edNCE. Since it is in S-HH (generated by G_6), it follows that B-edNCE $\subsetneqq$ C-edNCE, and so B-HH $\subsetneqq$ S-HH.

5. The Algebra of Hypergraphs with Ports

We define operations on abstract hypergraphs, thus making it possible to build "large" hypergraphs from "smaller" ones and to denote hypergraphs by algebraic expressions formed with these operations, together with constants denoting "elementary" hypergraphs. One of the operations is edge creation. In order to add a new edge to an abstract hypergraph one has to be able to refer to those vertices of the hypergraph that will be incident with the edge. It is for this reason that ports will be used (simply as reference points). Thus, in fact, we consider operations on abstract hypergraphs with ports.

We shall use the following new notation. For a hypergraph $H = (V_H, E_H, \text{port}_H)$ with ports, the *type* of H is its largest port number, i.e., the integer $\text{type}(H) = \max\{i \in \mathbb{N}_+ \mid \text{port}_H(i) \neq \varnothing\}$. The type of a hypergraph without ports is 0. Note that isomorphic hypergraphs have the same type; thus we define $\text{type}([H]) = \text{type}(H)$ for abstract hypergraphs. For $L \subseteq [\text{HG}^p(A)]$ we denote by $\text{maxtype}(L)$ the least upper bound of $\{\text{type}(H) \mid H \in L\}$.

The operations are first defined on concrete hypergraphs and then, in a standard way, extended to abstract ones. The first operation is the *disjoint union* of two hypergraphs H and H', denoted $H \oplus H'$. Let H and H' be two concrete disjoint hypergraphs with ports. We define $H \oplus H'$ to be the hypergraph with ports $(V_H \cup V_{H'}, E_H \cup E_{H'}, \text{port}_H \cup \text{port}_{H'})$. Note that $\text{type}(H \oplus H') = \max\{\text{type}(H), \text{type}(H')\}$. If H and H' are not disjoint, then $H \oplus H'$ is not defined.

If H_1 is isomorphic to H and H_1' to H', and if $H \oplus H'$ and $H_1 \oplus H_1'$ are both defined, then, clearly, $H \oplus H'$ and $H_1 \oplus H_1'$ are isomorphic. Thus, if H and H' are abstract hypergraphs with ports, then we define $H \oplus H' = [H_1 \oplus H_1']$, where H_1 and H_1' are disjoint, concrete hypergraphs, members of (the isomorphism classes) H and H', respectively. This uniquely defines $H \oplus H'$ for all abstract hypergraphs with ports.

The second operation is *edge creation*. This operation creates new edges between ports. For every $a \in A$, for every non-empty sequence $(i_1, ..., i_k)$ of positive integers, and for every hypergraph H in $\text{HG}^p(A)$, we define $\eta_{a, i_1, ..., i_k}(H) = K$, where $K = (V_H, E, \text{port}_H)$ with $E = E_H \cup \{(a, v_1, ..., v_k) \mid (i_j, v_j) \in \text{port}_H \text{ for every } j \in [1, k]\}$.

This operation applies to abstract hypergraphs in an obvious way $\eta_{a, i_1, \ldots, i_k}([H]) = [\eta_{a, i_1, \ldots, i_k}(H)]$.

Note that $\text{type}(K) = \text{type}(H)$. Also note that $K = H$ if H has no i_j-port for some $j \in [1, k]$. The above operation adds simultaneously to H at most $\text{card}(\text{port}_H(i_1)) \times \cdots \times \text{card}(\text{port}_H(i_k))$ edges labeled by a. "At most" and not always "exactly," because the hypergraphs do not have multiple edges with the same label; if an edge of the desired form already exists in H, it is not added. In particular, the operation $\eta_{a, i_1, \ldots, i_k}$ is idempotent. Note finally that an operation like $\eta_{a, 1, 1}$ establishes, in particular, all loops (a, v, v), where v is a 1-port.

The third and last operation is *port selection.* This operation is needed to be able to redefine the port numbers of vertices. If z is a finite subset of $\mathbb{N}_+ \times \mathbb{N}_+$ and $H \in \text{HG}^p(A)$, then we denote by $\pi_z(H)$ the hypergraph (V_H, E_H, port), where $\text{port} = \text{port}_H \circ z$ (in other words, $\text{port}(i) = \bigcup \{\text{port}_H(j) \mid (i, j) \in z\}$). The interpretation of a pair $(i, j) \in z$ is that the j-ports of H turn into i-ports of $\pi_z(H)$. Thus, the new ports are selected among the old ones. The operation of port selection applies to abstract hypergraphs in the obvious way: $\pi_z([H]) = [\pi_z(H)]$. Note that, in particular, $\pi_\varnothing(H)$ is the hypergraph without ports naturally associated with H. Note also that $\text{type}(\pi_z(H)) = \max\{i \mid \text{port}_H(j) \neq \varnothing \text{ for some } j \text{ with } (i, j) \in z\}$.

We finally introduce a few elementary hypergraphs; they will be used as basic objects in expressing hypergraphs by algebraic expressions. We denote by **0** the empty hypergraph (i.e., $V_0 = \varnothing$). We denote by **1** the abstract hypergraph (with ports) $[(V, E, \text{port})]$ with $V = \{v\}$, $E = \varnothing$, and $\text{port} = \{(1, v)\}$.

This ends our definitions of operations on hypergraphs with ports.

EXAMPLE 5.1. Let H be the abstract hypergraph with ports $[(V, E, \text{port})]$ with $V = \{v, w\}$, $E = \{(b, v, w)\}$, and $\text{port} = \{(1, v), (1, w)\}$, i.e., H consists of one b-labeled edge from v to w, and both v and w are 1-ports. Then $\pi_{\{(2,1)\}}(H) = [(V, E, \text{port}')]$ with $\text{port}' = \{(2, v), (2, w)\}$, and $\eta_{a, 1, 2}(H \oplus \pi_{\{(2,1)\}}(H))$ is the hypergraph shown in Fig. 23.

The reader will have noted that we have not defined three operations, but infinitely many. This makes $[\text{HG}^p(A)]$ into an F_A-algebra, where F_A is the infinite signature formally defined below.

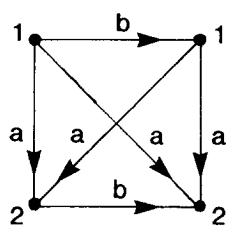

FIG. 23. A hypergraph with ports of type 2.

DEFINITION 5.2. F_A is the (one-sort) infinite signature consisting of

— the constants **0** and **1**,

— the binary symbol $\oplus$, and

— the unary symbols $\eta_{a, i_1, \ldots, i_k}$ (for all $a \in A$, $k \in \mathbb{N}_+$, and $i_1, \ldots, i_k \in \mathbb{N}_+$) and π_z (for all finite $z \subseteq \mathbb{N}_+ \times \mathbb{N}_+$).

It follows that the set $[\mathrm{HG}^p(A)]$ of abstract hypergraphs with ports is turned into an F_A-algebra, by the definitions of the operations of disjoint union, edge creation, and port selection, and of the constants **0** and **1**. We denote by $T(F_A)$ the set of well-formed terms over F_A. Each of these terms, say t, is called a *hypergraph expression*, or just an *expression*, and denotes an (abstract) hypergraph $\mathrm{val}(t)$ in $[\mathrm{HG}^p(A)]$: the *value* of t; $\mathrm{val}(t)$ is defined in the usual way, by induction on the structure of t, as follows (the fact that the symbols in F_A also denote operations on $[\mathrm{HG}^p(A)]$ should not confuse the reader):

$$\mathrm{val}(\mathbf{0}) = \mathbf{0}, \qquad \mathrm{val}(\mathbf{1}) = \mathbf{1}, \qquad \mathrm{val}(t \oplus t') = \mathrm{val}(t) \oplus \mathrm{val}(t'),$$

$$\mathrm{val}(\pi_z(t)) = \pi_z(\mathrm{val}(t)), \qquad \mathrm{val}(\eta_{a, i_1, \ldots, i_k}(t)) = \eta_{a, i_1, \ldots, i_k}(\mathrm{val}(t)).$$

In fact, val is the unique homomorphism from $T(F_A)$ to $[\mathrm{HG}^p(A)]$.

For every $h \in \mathbb{N}_+$, we let $F_{A,h}$ be the finite subset of F_A consisting of:

— the constants **0** and **1**,

— the binary symbol $\oplus$,

— the unary symbols $\eta_{a, i_1, \ldots, i_k}$ with k, $i_1, \ldots, i_k \in [1, h]$, and π_z with $z \subseteq [1, h] \times [1, h]$.

Intuitively this means that we restrict attention to hyperedges of rank $\leqslant h$, and to port numbers $\leqslant h$. We define the *width* of an expression t, denoted by $\mathrm{width}(t)$, to be the smallest h such that $t \in T(F_{A,h})$, where $T(F_{A,h})$ is the set of all terms over the signature $F_{A,h}$. Every expression has a finite width. The width of $H \in [\mathrm{HG}^p(A)]$ is defined as $\mathrm{width}(H) = \min\{\mathrm{width}(t) \mid \mathrm{val}(t) = H\}$. It should be clear that $\mathrm{type}(H) \leqslant \mathrm{width}(H)$; formally, it can be shown by induction on the structure of t that $\mathrm{type}(\mathrm{val}(t)) \leqslant \mathrm{width}(t)$, using previous remarks on "type." The following proposition shows that $\mathrm{width}(H)$ is finite.

PROPOSITION 5.3. *Every (abstract) hypergraph with ports is the value of an expression.*

Proof. Let $H = (V, E, \mathrm{port})$ be a hypergraph in $\mathrm{HG}^p(A)$. We explain how an expression can be constructed with value $[H]$. Example 5.4(1) below shows the result of this construction for a small hypergraph.

If $V = \varnothing$ then $\mathrm{val}(\mathbf{0}) = [H]$. Otherwise, we may assume that $V = [1, m]$ for some positive integer m. For every i in V, let $t_i = \pi_{\{(i,1)\}}(\mathbf{1})$, and consider the expression $t' = t_1 \oplus t_2 \oplus \cdots \oplus t_m$. Clearly, $\mathrm{val}(t') = [H']$, where $H' \in \mathrm{HG}^p(A)$ is a discrete

hypergraph with the same vertices as H, such that vertex i is the unique i-port of H'. In order to obtain H we add the edges, one by one. The edge $(a, i_1, ..., i_k)$ of H is added to H' by means of the operation $\eta_{a, i_1, ..., i_k}$. Thus, an expression t'' is obtained, of the form $\eta_{a, i_1, ..., i_k}(\cdots(\eta_{b, j_1, ..., j_n}(t'))\cdots)$, that denotes $[H]$, up to the ports. The desired expression is then $\pi_z(t'')$, where $z = \text{port}_H$. ▮

EXAMPLE 5.4. (1) The hypergraph shown in Fig. 23 is of type 2. Let us number its vertices from 1 to 4, from left to right and top to bottom. The construction in the proof of Proposition 5.3 produces the expression

$$\pi_z(\eta_{a,2,4}(\eta_{a,2,3}(\eta_{a,1,4}(\eta_{a,1,3}(\eta_{b,3,4}(\eta_{b,1,2}(\pi_u(\mathbf{1}) \oplus \pi_v(\mathbf{1}) \oplus \pi_x(\mathbf{1}) \oplus \pi_y(\mathbf{1}))))))))),$$

where $z = \{(1,1), (1,2), (2,3), (2,4)\}$, $u = \{(1,1)\}$, $v = \{(2,1)\}$, $x = \{(3,1)\}$, and $y = \{(4,1)\}$. This expression has not been constructed economically. We have not used the full power of the edge creating operations: these operations can add to a hypergraph several edges simultaneously, whereas we have added the edges one by one. The following expression, of smaller width (2 instead of 4), defines the same hypergraph: $\eta_{a,1,2}(t \oplus \pi_v(t))$ with $t = \pi_w(\eta_{b,1,2}(\mathbf{1} \oplus \pi_v(\mathbf{1})))$, $v = \{(2,1)\}$, and $w = \{(1,1), (1,2)\}$. Note that val(t) is the hypergraph H of Example 5.1. Note also that this expression shows that width(K) = 2 (where K is the hypergraph of Fig. 23) because type(K) $\leqslant$ width(K).

(2) The complete graph K_n with n vertices and all possible edges linking every two, possibly equal, vertices (and all vertices and edges labeled by a) is denoted by the expression $\pi_{\varnothing}(\eta_{a,1}(\eta_{a,1,1}(\mathbf{1} \oplus \cdots \oplus \mathbf{1})))$, with n times $\mathbf{1}$. Note that $\eta_{a,1,1}$ establishes all edges, including loops, whereas $\eta_{a,1}$ establishes all vertex labels. This expression is of width 1, for all values of n. The hypergraphs without ports can be denoted by the expressions in [1]. A notion of width is also associated with these expressions. The width of K_n in the sense of [1] is $n+2$. This indicates that our operations are "more powerful" than those of [1]. This is due in particular to the edge creation operations that can add in one stroke an unbounded number of edges to a hypergraph, whereas with the operations of [1], edges can be added one by one only.

DEFINITION 5.5. Let $\mathcal{X}$ be the set of n variables $\{X_1, ..., X_n\}$. Let $T(F_A, \mathcal{X})$ denote the set of all well-formed terms over $F_A \cup \mathcal{X}$, also called *expressions with variables*. Every expression $t \in T(F_A, \mathcal{X})$ defines a mapping $\|t\|$ on subsets of $[\text{HG}^p(A)]$, $\|t\|: \mathcal{P}([\text{HG}^p(A)])^n \to \mathcal{P}([\text{HG}^p(A)])$. If $L_1, ..., L_n$ are sets of (abstract) hypergraphs with ports, $\|t\|(L_1, ..., L_n)$ is a set of abstract hypergraphs with ports, defined by induction on the structure of t as follows:

if $t = X_i$ then $\|t\|(L_1, ..., L_n) = L_i$,

if $t = \mathbf{0}$ then $\|t\|(L_1, ..., L_n) = \{\mathbf{0}\}$, and similarly for $\mathbf{1}$,

if $t = t' \oplus t''$ then $\|t\|(L_1, ..., L_n) = \{H' \oplus H'' \mid H' \in \|t'\|(L_1, ..., L_n),\ H'' \in \|t''\|(L_1, ..., L_n)\}$,

if $t=f(t')$, where f is a unary operation of F_A, then $\|t\|(L_1, ..., L_n) = \{f(H) \mid H \in \|t'\|(L_1, ..., L_n)\}$.

In fact, the operation $\|t\|$ is the derived operation of t in the subset algebra $\mathscr{P}([\mathrm{HG}^p(A)])$. Since we now have expressions to denote hypergraphs, we can use systems of equations to define sets of hypergraphs (cf. [36]). See [5] for a thorough study of polynomial systems and their least solutions in arbitrary algebras.

DEFINITION 5.6. Let A be an alphabet. A *system* S over F_A is a *polynomial system of equations* over the signature F_A, in the sense of [5]. That is, $S = \langle X_1 = p_1, X_2 = p_2, ..., X_n = p_n \rangle$ with $n \geqslant 1$, where each p_i is a *polynomial* of the form $p_i = m_{i,1} \cup m_{i,2} \cup \cdots \cup m_{i,r_i}$ (or of the degenerate form $\varnothing$), and each $m_{i,j}$ is a *monomial*, i.e., an expression in $T(F_A, \mathscr{X})$, where $\mathscr{X} = \{X_1, ..., X_n\}$. The set of variables $\mathscr{X}$ is called the set of *unknowns* of S.

A *solution* of S in $\mathscr{P}([\mathrm{HG}^p(A)])$ is an n-tuple $(L_1, ..., L_n)$ of subsets of $[\mathrm{HG}^p(A)]$ such that $L_i = \|p_i\|(L_1, ..., L_n)$ for every $i \in [1, n]$, where $\|p_i\|(L_1, ..., L_n) = \bigcup \{\|m_{i,j}\|(L_1, ..., L_n) \mid j \in [1, r_i]\}$ (or $\|p_i\|(L_1, ..., L_n) = \varnothing$ in case p_i is $\varnothing$). Every system has a least solution in $\mathscr{P}([\mathrm{HG}^p(A)])$, with respect to componentwise inclusion.

The *hypergraph language defined by* S, denoted $\mathrm{L}(S)$, is the first component of the least solution of S in $\mathscr{P}([\mathrm{HG}^p(A)])$. A set of hypergraphs with ports is *equational* if it is defined by some system.

One of our main results is that a set of hypergraphs (without ports) is equational if and only if it is generated by an S-HH grammar (see Theorem 6.1). This can be called a fixed point characterization of S-HH (because a solution of S is also said to be a fixed point of S).

EXAMPLE 5.7. (1) We first consider a polynomial system S_1 that defines the set of "ladders," generated by S-HH grammar G_1 (see Fig. 6). S_1 has two unknowns: X_1 and X_2. It has the following two equations:

$$X_1 = \pi_{\varnothing}(X_2) \qquad \text{and} \qquad X_2 = m_1 \cup m_2,$$

where

$$m_2 = \eta_{b,1,2}(\mathbf{1}_q \oplus \mathbf{2}_q) \qquad \text{and} \qquad m_1 = \pi_z(\eta_{c,2,4}(\eta_{b,3,4}(\eta_{a,3,1}(\mathbf{3}_q \oplus \mathbf{4}_q \oplus X_2))))$$

with $\mathbf{i}_q = \pi_{\{(i,1)\}}(\eta_{q,1}(\mathbf{1}))$ for $i = 1, ..., 4$, and $z = \{(1,3), (2,4)\}$. Note that the three monomials of S_1 closely correspond to the three productions of G_1. Intuitively, if (L_1, L_2) is the least solution of S_1, then L_1 is the set of all ladders, as in Fig. 7, whereas L_2 is the set of all ladders, as in Fig. 7, of which the two "left-most" vertices are ports: the "upper" one is a 1-port and the "lower" one is a 2-port.

It is well known that the system of inequalities $X_i \supseteq p_i$ has the same least solution

as the system of equations $X_i = p_i$. In the case of system S_1 this means that (L_1, L_2) is the least pair of sets satisfying

(a) $\eta_{b,1,2}(\mathbf{1}_q \oplus \mathbf{2}_q) \in L_2$,

(b) $\pi_z(\eta_{c,2,4}(\eta_{b,3,4}(\eta_{a,3,1}(\mathbf{3}_q \oplus \mathbf{4}_q \oplus L_2)))) \subseteq L_2$, and

(c) $\pi_{\varnothing}(L_2) \subseteq L_1$.

In words this can be read as the recursive definition of "ladder with ports" and "ladder":

(a) The graph with two vertices (both ports) and one b-labeled edge is a ladder with ports.

(b) If H is a ladder with ports, then so is the one that is obtained from H by adding one square to the left (and moving the port numbers appropriately).

(c) If H is a ladder with ports, then one obtains a ladder by dropping the port numbers from H.

Thus, polynomial systems correspond to the usual way of defining properties of graphs recursively (where, in our case, the allowable graph operations are restricted to those in F_A).

(2) The set of all complete graphs with loops is defined by the following system S_c (cf. Example 5.4(2)):

$$X_1 = \pi_{\varnothing}(\eta_{a,1}(\eta_{a,1,1}(X_2))) \quad \text{and} \quad X_2 = \mathbf{1} \cup (\mathbf{1} \oplus X_2).$$

Obviously, the second equation defines all discrete graphs (of which every vertex is a 1-port), whereas the first equation adds all edges (and labels).

The set of all complete graphs (without loops) is defined by the following system S_2 (cf. grammar G_2 in Fig. 8):

$$X_1 = \pi_{\varnothing}(X_2),$$

$$X_2 = \eta_{a,1}(\mathbf{1}) \cup \pi_{\{(1,2),(1,1)\}}(\eta_{a,2}(\eta_{a,2,1}(\eta_{a,1,2}(\pi_{\{(2,1)\}}(\mathbf{1} \oplus X_2))))).$$

Intuitively, X_2 defines all complete graphs (without loops) of which all vertices are 1-ports. The second monomial of the second equation expresses the following: if H is a complete graph, then another complete graph can be obtained from H by adding one vertex v and adding all edges from the vertices of H to v, and vice versa.

(3) The set of all co-graphs is defined by the following system S_3 (cf. grammar G_3 in Fig. 10):

$$X_1 = \pi_{\varnothing}(X_2) \quad \text{and}$$

$$X_2 = \eta_{a,1}(\mathbf{1}) \cup (X_2 \oplus X_2) \cup \pi_{\{(1,2),(1,1)\}}(\eta_{a,2,1}(\eta_{a,1,2}(X_2 \oplus \pi_{\{(2,1)\}}(X_2)))).$$

X_2 defines all co-graphs of which all vertices are 1-ports. The three monomials of the equation for X_2 correspond directly to the three parts of the recursive definition of co-graphs, as given in the examples in Section 2. In fact, the last monomial

expresses that if H_1 and H_2 are two co-graphs (of which all vertices are 1-ports), then so is $H_1 \times H_2$. In the monomial, $H_1 \times H_2$ is constructed by first changing all 1-ports of H_2 into 2-ports, taking their disjoint union, adding all edges between H_1 and H_2, and finally changing all 2-ports back into 1-ports.

Every system like S in Definition 5.6 can also be solved formally, that is, in $\mathscr{P}(T(F_A))$. Let $(T_1, ..., T_n)$ denote the corresponding least solution. Then, by Proposition 13.3 of [5], $L_i = \{\text{val}(t) \mid t \in T_i\}$ for every $i \in [1, n]$, where $(L_1, ..., L_n)$ is the least solution of S in $\mathscr{P}([\text{HG}^p(A)])$. Moreover, by Proposition 13.5 of [5], $T_i = \{t \in T(F_A) \mid X_i \overset{*}{\underset{S'}{\Rightarrow}} t\}$, where S' is the ground rewriting system (or context-free grammar) on $T(F_A, \mathscr{X})$ consisting of the rules $X_i \to m_{i,j}$ for all $i \in [1, n]$ and $j \in [1, r_i]$. Taking these facts together, we obtain the following proposition.

PROPOSITION 5.8. *If $(L_1, ..., L_n)$ is the least solution of S in $\mathscr{P}([\text{HG}^p(A)])$, then, for every $i \in [1, n]$, $L_i = \{\text{val}(t) \mid t \in T(F_A), X_i \overset{*}{\underset{S'}{\Rightarrow}} t\}$.*

Hence, clearly, we have the following result.

LEMMA 5.9. *The hypergraphs belonging to the least solution of a system S are of width at most h, where h is the smallest integer such that all monomials of S are in $T(F_{A,h}, \mathscr{X})$.*

Our next aim is to characterize in terms of graph grammars the equational sets of hypergraphs (with and without ports). In order to do so, we shall interpret every monomial $m_{i,j}$ (of the equation $X_i = p_i$ of S, see Definition 5.6) as a hypergraph with ports $H_{i,j}$ that will be the right-hand side of a graph rewriting rule of the form $X_i \to H_{i,j}$. There is one difficulty. Consider, e.g., the monomial $\eta_{a,1,2}(\mathbf{1} \oplus X)$, with $X \in \mathscr{X}$. Since X stands for hypergraphs of unbounded type, the "natural" representation of this monomial is a hypergraph with a hyperedge of infinite rank, labeled X, see Fig. 24. However, by Lemma 5.9 (and the fact that $\text{type}(H) \leqslant \text{width}(H)$), the hypergraphs of an equational set are of bounded type. Hence the unknowns can be typed *a posteriori*, and the monomials over "typed" unknowns can be represented by hypergraphs in $\text{HG}^p(A \cup \mathscr{X})$. The integer h of Lemma 5.9 gives a rough upper

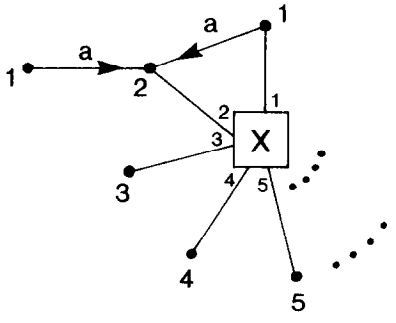

FIG. 24. An impossible hypergraph.

bound of maxtype(L), where $L = \mathrm{L}(S)$. In fact, the number maxtype(L) can be computed from S. For $H \in [\mathrm{HG}^p(A)]$ we define realport(H) $= \{i \mid \mathrm{port}_H(i) \neq \varnothing\}$, and for $L \subseteq [\mathrm{HG}^p(A)]$, realport($L$) $= \{\mathrm{realport}(H) \mid H \in L\}$. Using the iterative technique on page 104 of [5] it can be shown that realport(L) can be computed from S. It is then easy to obtain maxtype(L) from realport(L).

We introduce expressions and systems of a special form.

DEFINITION 5.10. Let $\mathcal{X}$ be a set of variables. An expression t in $T(F_A, \mathcal{X})$ is *guarded* if each occurrence of a variable X in t is in a subexpression of t of the form $\pi_z(X)$. A polynomial system over F_A is *guarded* if its monomials are guarded expressions in $T(F_A, \mathcal{X})$, where $\mathcal{X}$ is the set of unknowns of the system.

With every guarded expression $t \in T(F_A, \mathcal{X})$ we associate a hypergraph in $[\mathrm{HG}^p(A \cup \mathcal{X})]$ that we call its *value* and that we denote by val(t). It is defined by induction on the structure of t, as in the case where t is in $T(F_A)$ (see the definition of val(t) after Definition 5.2), with the following additional clause: for $X \in \mathcal{X}$, $\mathrm{val}(\pi_z(X)) = [H]$, where H is the concrete hypergraph with ports such that $V_H = [1, k]$ with $k = \max(\{1\} \cup \{j \mid (i, j) \in z \text{ for some } i \in \mathbb{N}_+\})$, $E_H = \{(X, 1, 2, \ldots, k)\}$, and $\mathrm{port}_H = z$. H will be called the concrete value of $\pi_z(X)$, denoted $\mathrm{cval}(\pi_z(X))$.

The value of the guarded expression $\eta_{a,1,2}(\mathbf{1} \oplus \pi_z(X))$, where $z = \{(2, 3), (3, 2), (3, 1)\}$ is shown in Fig. 25, cf. Fig. 24.

In the following characterization of the hypergraphs defined by guarded expressions, we say that an edge is *nonterminal* if it is labeled by a symbol in $\mathcal{X}$, and *terminal* otherwise (as motivated by the correspondence between guarded systems and S-HH grammars that we shall establish in the next section). Note that for hypergraphs with nonterminal and terminal edges the notions "separated" and "loop-free" are defined in Section 2.

LEMMA 5.11. *A hypergraph H in $[\mathrm{HG}^p(A \cup \mathcal{X})]$ is* val(t) *for some guarded expression t in $T(F_A, \mathcal{X})$ if and only if H is separated and loop-free.*

Proof. The "only-if" part is by induction on the structure of t: it suffices to note that the separated and loop-free properties are preserved by the operations of

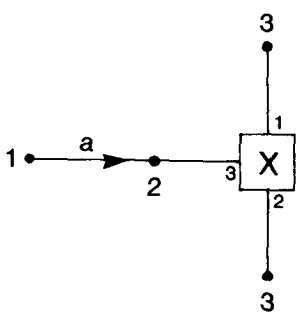

FIG. 25. The value of a guarded expression.

disjoint union, edge creation (of terminal edges!), and port selection, and that they hold for the base cases $\mathbf{0}$, $\mathbf{1}$, and $\pi_z(X)$.

The "if" part can be proved by an easy adaptation of the proof of Proposition 5.3. The terminal vertices, terminal edges, and port numbers are introduced as discussed in that proof. A nonterminal edge $(X, i_1, ..., i_k)$, with its incident vertices $i_1, ..., i_k$, is introduced by a base subexpression of the form $\pi_z(X)$, where $z = \{(i_j, j) \mid j \in [1, k]\}$. ∎

EXAMPLE 5.12. Let H be the hypergraph with ports that is the right-hand side of the second production of G_1 in Fig. 6. Then H is the value of the guarded expression $\pi_z(\eta_{c,2,4}(\eta_{b,3,4}(\eta_{a,3,1}(\mathbf{3}_q \oplus \mathbf{4}_q \oplus \pi_x(X_2)))))$, where $\mathbf{3}_q = \pi_{\{(3,1)\}}(\eta_{q,1}(\mathbf{1}))$, $\mathbf{4}_q = \pi_{\{(4,1)\}}(\eta_{q,1}(\mathbf{1}))$, $x = \{(1,1), (2,2)\}$, and $z = \{(1,3), (2,4)\}$. Note that this expression equals m_1 in Example 5.7(1), with X_2 replaced by $\pi_x(X_2)$.

The next (key) lemma shows that the substitution of guarded expressions for variables in guarded expressions corresponds (by the mapping val) to the substitution of hypergraphs for edges in hypergraphs, as given in Definition 2.1.

LEMMA 5.13. *Let t be a guarded expression in $T(F_A, \mathcal{X})$, and let K be a concrete hypergraph in $\mathrm{HG}^p(A \cup \mathcal{X})$ such that $\mathrm{val}(t) = [K]$. Then there is a bijection between the occurrences of variables in t and the nonterminal edges of K, such that, for every $X \in \mathcal{X}$,*

(1) *occurrences of the variable X correspond to nonterminal edges labeled X,*

(2) *for every guarded expression s in $T(F_A, \mathcal{X})$ and every concrete hypergraph H in $\mathrm{HG}^p(A \cup \mathcal{X})$, disjoint with K, such that $\mathrm{val}(s) = [H]$, if t' is the guarded expression obtained from t by the substitution of s for an occurrence of X and e is the nonterminal edge corresponding to that occurrence, then $\mathrm{val}(t') = [K[H/e]]$ (i.e., $\mathrm{val}(t')$ is the abstract hypergraph corresponding to the concrete hypergraph $K[H/e]$).*

Proof. For fixed t, the property of K stated above is preserved by isomorphism, i.e., if K' is isomorphic to K then K' has the property too. In fact, the bijection between the occurrences of variables in t and the nonterminal edges of K can be combined, in an obvious way, with the isomorphism between K' and K to obtain a similar bijection for K'. To see that (2) holds for K', note that if H' is isomorphic to H and e' corresponds to e through the isomorphism between K' and K, then $K'[H'/e']$ is isomorphic to $K[H/e]$.

Thus, to prove the lemma, it suffices to show for every t the existence of one K with $\mathrm{val}(t) = [K]$ and one bijection satisfying (1) and (2). This will be done by induction on the structure of t.

The only interesting base case is $t = \pi_z(X)$. Take $K = \mathrm{cval}(\pi_z(X))$. Obviously, the unique occurrence of X in t is in bijection with the unique (X-labeled) edge e of K. Now consider $t' = \pi_z(s)$. Clearly, $\mathrm{val}(t') = [\pi_z(H)]$. Thus, it suffices to prove that

$\pi_z(H) = \mathrm{cval}(\pi_z(X))[H/e]$. We need only compare the ports of these two hypergraphs. For the first one, by the definition of π_z, port $= \{(i, v) \mid (i, j) \in z$ and $(j, v) \in \mathrm{port}_H\}$. For the second one, by Definition 2.1 and the definition of $\mathrm{cval}(\pi_z(X))$, port $= \{(i, v) \mid (i, j) \in z$ and $v \in \mathrm{rep}(j)\}$. Since $\mathrm{rep}(j) = \mathrm{port}_H(j)$, the two sets of ports are equal.

We now consider the inductive cases. Let us first consider the case that $t = \eta_{a, i_1, \ldots, i_k}(t_1)$. Let K_1 be associated by induction to t_1 (and so $\mathrm{val}(t_1) = [K_1]$) and take $K = \eta_{a, i_1, \ldots, i_k}(K_1)$. In an obvious way, the bijection between t_1 and K_1 carries over to t and K. Consider $t' = \eta_{a, i_1, \ldots, i_k}(t_1')$. For some edge e of K_1 we have $\mathrm{val}(t_1') = [K_1[H/e]]$ by induction hypothesis. It remains to show that $\mathrm{val}(t') = [\eta_{a, i_1, \ldots, i_k}(K_1)[H/e]]$, or, that edge creation commutes with hypergraph substitution:

$$\eta_{a, i_1, \ldots, i_k}(K_1[H/e]) = \eta_{a, i_1, \ldots, i_k}(K_1)[H/e].$$

Let us denote these two (concrete) hypergraphs by K_2 and K_3. Since the edge creation operation does not modify the existing vertices, edges, and ports, K_2 and K_3 have the same vertices, the same ports, and they have in common the edges of $K_1[H/e]$. We need only compare the edges created in K_2 and K_3 by the operation $\eta_{a, i_1, \ldots, i_k}$. Let $(a, w_1, \ldots, w_k)$ be such an edge in K_2. Then w_j is an i_j-port of $K_1[H/e]$. By Definition 2.1, w_j belongs to $\mathrm{rep}(v_j)$, where v_j is an i_j-port of K_1. Consider the edge $(a, v_1, \ldots, v_k)$ of $\eta_{a, i_1, \ldots, i_k}(K_1)$. By Definition 2.1 it yields the edge $(a, w_1, \ldots, w_k)$ of $\eta_{a, i_1, \ldots, i_k}(K_1)[H/e]$, i.e., of K_3. Conversely, an edge of K_3, coming from an edge of $\eta_{a, i_1, \ldots, i_k}(K_1)$ of the form $(a, v_1, \ldots, v_k)$, where v_j is an i_j-port of K_1, is of the form $(a, w_1, \ldots, w_k)$ as above; hence it is in K_2. Hence $K_2 = K_3$.

We now consider similarly the case that $t = \pi_z(t_1)$, and we need only establish that port selection commutes with hypergraph substitution:

$$\pi_z(K_1[H/e]) = \pi_z(K_1)[H/e].$$

Let K_2 and K_3 be the left-hand and right-hand sides of this equality, respectively. We need only compare their ports. Let w be an i-port of K_2. Then w is a j-port of $K_1[H/e]$ for some $(i, j) \in z$. Hence $w \in \mathrm{rep}(v)$, where v is a j-port of K_1. It follows that v is an i-port of $\pi_z(K_1)$ and that w is an i-port of $\pi_z(K_1)[H/e] = K_3$. The proof in the other direction is similar.

We finally consider the case that $t = t_1 \oplus t_2$. By induction, let K_1 and K_2 be associated with t_1 and t_2, respectively. Since the required property is preserved by isomorphism (as observed above), we may assume that K_1 and K_2 are disjoint. Take $K = K_1 \oplus K_2$. In the obvious way, the bijections between t_1 and K_1, and between t_2 and K_2, can be combined into a bijection between $t_1 \oplus t_2$ and $K_1 \oplus K_2$. Without loss of generality we assume that the occurrence of X is in t_1 and that the corresponding edge e is in K_1. We obtain the desired result because disjoint union commutes with hypergraph substitution: $K_1[H/e] \oplus K_2 = (K_1 \oplus K_2)[H/e]$. ∎

6. Separated Handle Hypergraph Grammars and Systems of Equations

Our second main result is an algebraic fixed-point characterization of the hypergraph languages generated by S-HH grammars: they are exactly the equational sets of hypergraphs.

THEOREM 6.1. S-HH $= \{\mathrm{L}(S) \mid S$ *is a polynomial system with* $\mathrm{L}(S) \subseteq [\mathrm{HG}(A)]$ *for some* $A\}$.

This theorem follows directly from the next proposition.

PROPOSITION 6.2. *For a subset* L *of* $[\mathrm{HG}(A)]$, *the following are equivalent*:

(1) L *is defined by a polynomial system over* F_A.

(2) L *is defined by a guarded polynomial system over* F_A.

(3) L *is generated by an* S-HH *grammar with terminal alphabet* A.

The proof of this proposition will be given after those of two additional lemmas. We shall first establish the equivalence of (1) and (2), and then the equivalence of (2) and (3).

Let S be a system with set of unknowns $\mathscr{X} = \{X_1, ..., X_n\}$ and the least solution $(L_1, ..., L_n)$. Let h_i be an integer such that $\mathrm{maxtype}(L_i) \leqslant h_i$ for all i (one can take for h_i the integer h defined in Lemma 5.9, or one can take $h_i = \mathrm{maxtype}(L_i)$). Let $z_i = \{(j, j) \mid j \in [1, h_i]\}$.

For every expression t in $T(F_A, \mathscr{X})$ we denote by $\bar{t}$ the guarded expression obtained by substituting in t every occurrence of a variable X_i, $i \in [1, n]$, by the expression $\pi_{z_i}(X_i)$. We denote by $\bar{S}$ the guarded system obtained from S by the replacement of every monomial t by $\bar{t}$. The next lemma establishes that (1) implies (2) in Proposition 6.2.

LEMMA 6.3. *The systems* S *and* $\bar{S}$ *have the same least solution in* $\mathscr{P}([\mathrm{HG}^p(A)])$.

Proof. If H is a hypergraph in $[\mathrm{HG}^p(A)]$ of type at most h_i, then $\pi_{z_i}(H) = H$. It follows that, if $t \in T(F_A, \mathscr{X})$ and $M_1, ..., M_n$ are sets of hypergraphs in $[\mathrm{HG}^p(A)]$ of maxtypes at most $h_1, ..., h_n$, then $\|t\|(M_1, ..., M_n) = \|\bar{t}\|(M_1, ..., M_n)$. This can be shown by an easy induction on the structure of t, using the above remark and Definition 5.5. Since the least solution $(L_1, ..., L_n)$ of S in $\mathscr{P}([\mathrm{HG}^p(A)])$ is such that $\mathrm{maxtype}(L_i) \leqslant h_i$, this n-tuple is also a solution of $\bar{S}$. Hence $\bar{L}_i \subseteq L_i$, where $(\bar{L}_1, ..., \bar{L}_n)$ denotes the least solution of $\bar{S}$. It follows that all hypergraphs in $\bar{L}_i$ are of type at most h_i, and that $\|t\|(\bar{L}_1, ..., \bar{L}_n) = \|\bar{t}\|(\bar{L}_1, ..., \bar{L}_n)$ for every expression t. Hence, since $(\bar{L}_1, ..., \bar{L}_n)$ is a solution of $\bar{S}$, it is also a solution of S. Hence $L_i \subseteq \bar{L}_i$ for all $i \in [1, n]$, and finally $L_i = \bar{L}_i$, as was to be proved. ∎

EXAMPLE 6.4. Consider Example 5.7. For S_1, clearly, $\mathrm{maxtype}(L_1) = 0$ and $\mathrm{maxtype}(L_2) = 2$. Hence the equations of $\bar{S}_1$ are obtained from those of S_1 by

changing X_2 into $\pi_{\{(1,1),(2,2)\}}(X_2)$ in their right-hand sides (cf. Example 5.12). For all systems S_c, S_2, and S_3, maxtype$(L_1)=0$ and maxtype$(L_2)=1$. Hence, the guarded systems $\bar{S}_c$, $\bar{S}_2$, and $\bar{S}_3$ are obtained by changing X_2 into $\pi_{\{(1,1)\}}(X_2)$ in the right-hand sides of their equations.

We now compare S-HH grammars and guarded systems. Let $S=\langle X_1=p_1, ..., X_n=p_n\rangle$ be a guarded system over F_A, with $\mathcal{X}=\{X_1, ..., X_n\}$, and let $G=(N, A, P, X_{in})$ be a HH grammar. We say that S and G are *associated* if $N=\mathcal{X}$, $X_{in}=X_1$, and $[P]=\{X_i\to H \mid i\in[1,n],\ \text{val}(m)=[H]$ for some monomial m of $p_i\}$. Obviously, for every system S there is an HH grammar G associated with it. System $\bar{S}_1$ and grammar G_1 are associated (cf. Example 5.12). Grammar G_2 (with X replaced by X_1) is associated with the system that has one equation, viz. the second equation of system $\bar{S}_2$, with X_2 replaced by X_1. The same is true for G_3 and $\bar{S}_3$.

Let $(L_1, ..., L_n)$ be the least solution of S in $\mathcal{P}([\mathrm{HG}^p(A)])$. To obtain a relationship between S and G, we wish to view $L_1, ..., L_n$ as the hypergraph languages generated by $X_1, ..., X_n$. However, since the hypergraphs in these languages have ports, we have to extend the derivation relation of G. In Section 2, the derivation relation $K \underset{(e,p)}{\Longrightarrow} K'$ of G has been defined for hypergraphs K and K' and not for hypergraphs with ports. But its extension to hypergraphs with ports is straightforward because substitution has been defined in Definition 2.1 for hypergraphs with ports. Now we extend the definition of $\mathrm{L}(G)$ as follows: for K in $\mathrm{HG}^p(N\cup A)$, $\mathrm{L}(G,K)=\{[K'] \mid K'\in\mathrm{HG}^p(A),\ K\overset{*}{\Rightarrow}K'\}$. From Definition 2.1 it follows that $\text{type}(K')\leqslant\text{type}(K)$ if $K, K'\in\mathrm{HG}^p(A)$ and $K\underset{(e,p)}{\Longrightarrow}K'$ for some edge e of K and some production p of G. Hence the type of a hypergraph in $\mathrm{L}(G,K)$ is at most that of K.

Let $h\in\mathbb{N}$ and $z=\{(j,j)\mid j\in[1,h]\}$. We define $L(G,X_i,h)$ to be the set $L(G,K)$, where K is the hypergraph with ports that is the concrete value of the guarded expression $\pi_z(X_i)$, cf. Definition 5.10. Since, for $i=1$ and $h=0$, $\text{cval}(\pi_\varnothing(X_1))=\text{se}(X_1)$, it follows that $\mathrm{L}(G,X_1,0)=\mathrm{L}(G)$.

LEMMA 6.5. *Let S be a guarded system with unknowns $X_1, ..., X_n$ and least solution $(L_1, ..., L_n)$ in $\mathcal{P}([\mathrm{HG}^p(A)])$. Let G be an* HH *grammar associated with S:*

(1) *G is separated and loop-free.*

(2) *For every $i\in[1,n]$, if* maxtype$(L_i)\leqslant h$, *then* $\mathrm{L}(G,X_i,h)=L_i$.

Proof. (1) is an immediate consequence of Lemma 5.11.

(2) We know, from Proposition 5.8, that $L_i=\{\text{val}(t)\mid t\in T(F_A),\ X_i\overset{*}{\underset{S'}{\Rightarrow}}t\}$, where S' is constructed from S as explained just before Proposition 5.8. We first prove that $L_i\subseteq\mathrm{L}(G,X_i,h)$. Let $H\in L_i$. Then there is a rewriting sequence $X_i\overset{*}{\underset{S'}{\Rightarrow}}t$ such that $H=\text{val}(t)$. Hence, one has a rewriting sequence $\pi_z(X_i)\overset{*}{\underset{S'}{\Rightarrow}}\pi_z(t)$, where $z=\{(j,j)\mid j\in[1,h]\}$ as above. All expressions in this sequence are guarded. Hence, by Lemma 5.13 and the definition of G, one can obtain a (disjoint) derivation of the same length $K\overset{*}{\underset{G}{\Rightarrow}}K'$, where $K=\text{cval}(\pi_z(X_i))$ and $[K']=\text{val}(\pi_z(t))$.

Hence $\pi_z(H) = \mathrm{val}(\pi_z(t)) \in \mathrm{L}(G, X_i, h)$. But h was taken such that $\pi_z(H) = H$. Hence we have $L_i \subseteq \mathrm{L}(G, X_i, h)$ as desired.

Conversely, let $H \in \mathrm{L}(G, X_i, h)$. There is a disjoint derivation $\mathrm{cval}(\pi_z(X_i)) \overset{*}{\underset{G}{\Rightarrow}} K'$ with $[K'] = H$. By Lemma 5.13 there is a rewriting sequence of the same length $\pi_z(X_i) \overset{*}{\underset{S'}{\Rightarrow}} t$ such that $H = \mathrm{val}(t)$. Hence t is necessarily of the form $t = \pi_z(t')$ with $X_i \overset{*}{\underset{S'}{\Rightarrow}} t'$. Taking $H' = \mathrm{val}(t')$, we have $H' \in L_i$ and, by the choice of h, $\pi_z(H') = H'$. Since $H = \pi_z(H')$, we have $H \in L_i$. Hence we have established that $L_i = \mathrm{L}(G, X_i, h)$. ∎

Proof of Proposition 6.2. (1) → (2) follows from Lemma 6.3, and (2) → (1) holds trivially.

(2) → (3). Let $L \subseteq [\mathrm{HG}(A)]$ be $\mathrm{L}(S)$ for some guarded system S. Since $\mathrm{maxtype}(L) = 0$, it follows from Lemma 6.5 that $L = \mathrm{L}(G, X_1, 0)$ where G is any S-HH grammar associated with S. Hence $L = \mathrm{L}(G)$.

(3) → (2). Let $G = (\mathscr{X}, A, P, X_1)$ be an S-HH grammar. By Lemma 2.7 we may assume that G is loop-free. Since each right-hand side of a production of G is separated and loop-free, Lemma 5.11 implies that one can find a guarded system S, associated with G. Let $(L_1, ..., L_n)$ be its least solution. Lemma 6.5(2) shows that $L_1 = \mathrm{L}(G, X_1, h)$ where $h = \mathrm{maxtype}(L_1)$. Let S_0 be the system S augmented with the new equation $X_0 = \pi_\varnothing(X_1)$. Its least solution is $(L_0, L_1, ..., L_n)$ with $L_0 = \pi_\varnothing(L_1)$. Since $\pi_\varnothing(\mathrm{L}(G, X_1, h)) = \mathrm{L}(G, X_1, 0)$, it follows that $L_0 = \mathrm{L}(G, X_1, 0) = \mathrm{L}(G)$. Hence the set $\mathrm{L}(G)$ is defined by the guarded system S_0. ∎

From the S-HH grammar G_2 one obtains, according to the above proof, the system $\bar{S}_2$, defining the same language (and similarly, $\bar{S}_3$ is obtained from G_3).

It is left to the reader to show the following generalization of Theorem 6.1.

THEOREM 6.6. $\{\mathrm{L}(S) \mid S$ *is a polynomial system*$\} = \{\mathrm{L}(G, K) \mid G$ *is an* S-HH *grammar and* K *is a hypergraph with ports*$\}$.

Through Theorem 4.4, Theorem 6.1 also gives a characterization of the class C-edNCE. However, to characterize C-edNCE it would be more satisfactory (cf. Corollary 4.5) to restrict polynomial systems in such a way that they define graph languages only. Let $T_{\mathrm{gra}} \subseteq T(F_A, \mathscr{X})$ be the set of expressions with variables defined inductively as follows:

$\mathscr{X} \subseteq T_{\mathrm{gra}}$, $\mathbf{0} \in T_{\mathrm{gra}}$, $T_{\mathrm{gra}} \oplus T_{\mathrm{gra}} \subseteq T_{\mathrm{gra}}$,

$\eta_{a,1}(\mathbf{1}) \in T_{\mathrm{gra}}$ for all $a \in A$,

$\eta_{a,i_1,i_2}(T_{\mathrm{gra}}) \subseteq T_{\mathrm{gra}}$ for all $a \in A$ and $i_1, i_2 \in \mathbb{N}_+$,

$\pi_z(T_{\mathrm{gra}}) \subseteq T_{\mathrm{gra}}$ for all finite $z \subseteq \mathbb{N}_+ \times \mathbb{N}_+$.

For C-edNCE we obtain the following algebraic fixed-point characterization.

THEOREM 6.7. C-edNCE $= \{\mathrm{L}(S) \mid S$ *is a polynomial system satisfying* $P_{\mathrm{gra}}\}$, *where* P_{gra} *means that* S *has equation* $X_1 = \pi_\varnothing(X_2)$, *and* $m \in T_{\mathrm{gra}}$ *for every monomial* m *of* S.

Proof. The proof is easy, using Theorem 4.4, Corollary 4.5, and the obvious fact that (guarded) polynomial systems satisfying P_{gra} are associated with S-HH grammars satisfying P_{gra} (of Corollary 4.5) and vice versa. The details are left to the reader. ▮

In Example 5.7, S_1 and S_3 satisfy P_{gra}, but S_c and S_2 do not. Of course, if one would only be interested in defining graph languages, it would be more convenient to change the signature F_A by dropping **1**, dropping all $\eta_{a, i_1, \ldots, i_k}$ with $k \neq 2$, and introducing constants σ_a for every $a \in A$ with $\text{val}(\sigma_a)$ defined to be the graph $\text{val}(\eta_{a,1}(\mathbf{1}))$; cf. [12].

We end this section with a few easy consequences of Theorem 6.1.

THEOREM 6.8. *The set of hypergraphs in* $[\text{HG}(A)]$ *of width at most* h *is equational. Hence it is generated by an* S-HH *grammar.*

Proof. The hypergraphs in $[\text{HG}(A)]$ of width at most h are exactly those denoted by the expressions of the form $\pi_{\varnothing}(t)$, where $t \in T(F_{A,h})$. Recall that $F_{A,h}$ is a finite subset of F_A. It follows that the desired set is defined by the system of equations

$$X = \pi_{\varnothing}(Y), \qquad Y = (Y \oplus Y) \cup \mathbf{0} \cup \mathbf{1} \cup f_1(Y) \cup \cdots \cup f_m(Y),$$

where $\{f_1, \ldots, f_m\}$ is the set of unary symbols of $F_{A,h}$.

By Theorem 6.1 an S-HH grammar can be obtained from this system. We show in Fig. 26 a few productions of this grammar in the case that $h = 2$ and $A = \{a\}$. The fifth and sixth production correspond to the monomials $\eta_{a,1,1}(Y)$ and $\eta_{a,1,2}(Y)$, respectively. There are two other productions obtained from these two by exchanging 1 and 2 (as port numbers and ranks of nonterminal vertices). There are $2^4 = 16$ productions corresponding to the monomials $\pi_z(Y)$, where $z \subseteq [1,2] \times [1,2]$. The last two productions shown in Fig. 26 correspond to $z = \{(1,1), (1,2), (2,2)\}$ and $z = \{(1,2), (2,1)\}$, respectively. ▮

The results of [6, 7, 9, 10] on graph properties expressible in monadic second-order logic can be adapted in a straightforward way to the equational sets of hypergraphs with ports. By Theorem 6.1 these results apply to the class S-HH and by Theorem 4.4 to the class C-edNCE.

A concrete hypergraph with ports $H = (V, E, \text{port})$ can be completely described by the relational structure $|H| = (V, (\text{edg}_{a,k})_{a \in A, k \in \mathbb{N}_+}, (\text{pt}_i)_{i \in \mathbb{N}_+})$ where $\text{edg}_{a,k}(v_1, \ldots, v_k)$ is true iff $(a, v_1, \ldots, v_k) \in E$, and $\text{pt}_i(v)$ is true iff $(i, v) \in \text{port}$. We will use H to denote $|H|$. Monadic second-order formulas can thus be written, with quantifications over vertices and sets of vertices, and properties of hypergraphs can be expressed. Typical examples of such monadic second-order properties are vertex k-colourability and connectivity. We refer the reader to [7, 8, 9, 10] for more details. The adaptation to the present situation is straightforward. Let us observe that we do not allow quantification over edges and sets of edges. It follows that the

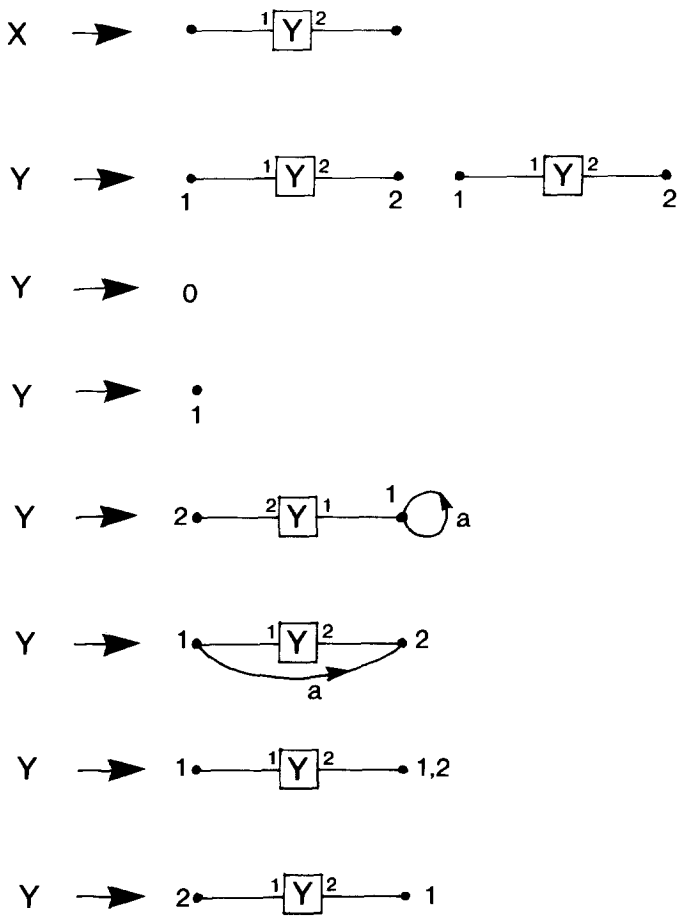

FIG. 26. Part of an S-HH grammar generating all hypergraphs of width ≤ 2.

existence of a Hamiltonian path is not expressible in the language we consider here, whereas it is in those that are constructed in [7, 9, 10]; see [8].

THEOREM 6.9. *Let* $L \subseteq [\mathrm{HG}(A)]$ *be generated by an* S-HH *grammar* G. *Let* ϕ *be a closed monadic second-order formula.*

(1) *One can construct from* G *and* ϕ *an* S-HH *grammar generating* $\{H \in L \mid H$ *satisfies* $\phi\}$.

(2) *One can decide whether or not all hypergraphs in* L *satisfy* ϕ.

(3) *Given* H *in* L *and a derivation of* H *in* G *of length* d, *one can decide in time* $O(d)$ *whether or not* H *satisfies* ϕ (*where the constant in* $O(d)$ *depends on* G *and* ϕ).

The same assertions hold for $L \subseteq [\mathrm{GR}(A)]$ *and* "C-edNCE" *instead of* "S-HH."

We will not prove this result. Similar results have been proved in detail in [6, 10], for different types of graph grammar. We refer the reader to their proofs; see also [11]. Together with Theorem 6.8 it gives the following corollary.

COROLLARY 6.10. *The set of hypergraphs in* [HG(A)] *of width at most h and that satisfy a monadic second-order formula can be generated by an* S-HH *grammar.*

7. COMPARISON WITH OTHER TYPES OF GRAPH GRAMMARS

In this section we show that the class S-HH includes the class C-NLC of graph languages generated by confluent NLC grammars [6]. We establish that it does not contain all hypergraph languages generated by the context-free hypergraph grammars (or hyperedge replacement grammars) of [1, 26, 27, 28]. Nevertheless, by a construction of [25], Theorem 6.9 is a common extension of the analogous results proved in [6, 10] for C-NLC and CFHG, respectively.

DEFINITION 7.1. An edNCE grammar $G = (N, A, P, X_{\text{in}})$ is an *NLC grammar* if

(1) $A = A' \cup \{*\}$, where $* \notin A'$, and, in every right-hand side of a production, every edge has label $*$, and every vertex has a label in A';

(2) for every right-hand side of a production, if $(*, u, v)$ is an edge, then so is $(*, v, u)$;

(3) there is a relation $\text{conn} \subseteq (N \cup A') \times (N \cup A')$, called the connection relation of G, such that, for every production $X \to (H, \text{emb})$ in P, $\text{emb} = \{(u, *, *, q, d) \mid u \in V_H, (\text{lab}_H(u), q) \in \text{conn}, d \in \{\text{in}, \text{out}\}\}$.

Requirements (1) and (2) just mean that the grammar generates undirected graphs of which the vertices are labeled, but not the edges. Requirement (3) means that the embedding is "node label controlled."

Let C-NLC denote the class of graph languages generated by confluent NLC grammars. Thus, by definition, C-NLC $\subseteq$ C-edNCE (and the inclusion is proper: $L(G_9)$ is a counterexample, see [24]). It follows that Theorem 6.9 (for C-edNCE) entails the corresponding result for C-NLC established in [6] (to see this for Theorem 6.9(1) one has to know that the constructions in [6, 10] preserve most of the structure of the grammar). To be more precise, the monadic second-order language in [6] is slightly different from the one in this paper; it has atomic formulas $\text{edg}(v_1, v_2)$, meaning that there is an undirected edge between v_1 and v_2, and $\text{lab}_a(v)$, meaning that v has label a. However, it is easy to see that every formula ϕ in [6] can be translated into a formula ϕ' of our language, such that for every undirected, vertex-labeled graph H, H satisfies ϕ iff H satisfies ϕ'; just translate $\text{edg}(v_1, v_2)$ into $\text{edg}_{*,2}(v_1, v_2)$, and $\text{lab}_a(v)$ into $\text{edg}_{a,1}(v)$.

Remarks 7.2. Some confluent NLC grammars are not associative (see [6]) but their translation into C-edNCE grammars always yields associative grammars by Lemma 3.2. We shall explain this apparent contradiction.

Let G be an NLC grammar with connection relation conn. For an undirected, vertex-labeled graph H, let $m(H)$ denote the graph with embedding (H, emb), where emb is defined in terms of conn, as in Definition 7.1(3).

Now, if K and H are undirected, vertex-labeled graphs and v is a vertex of K, then the graph with embedding $m(K[m(H)/v])$ is not necessarily equal to $m(K)[m(H)/v]$ (they have the same vertices and edges, but not necessarily the same embedding). Hence, if M is another graph and w is a vertex of M, then

(i) $m(M)[m(K)/w][m(H)/v] = m(M)[m(K)[m(H)/v]/w]$ without having necessarily

(ii) $M[m(K)/w][m(H)/v] = M[m(K[m(H)/v])/w]$.

As an example (cf. p. 163 of [6]), let $A = \{a, b, c\}$, let $\text{conn} = \{b, c\} \times A$, and let $K = b - X$ (a graph with two vertices labeled by b and X and connected by an edge), and $H = c$ (a graph with one vertex labeled by c). If we denote by y any node labeled by y (for all y), then the embedding relation of $m(H)$ is $\{(c, *, *, q, d) \mid q \in A,\ d \in \{\text{in}, \text{out}\}\}$, and hence $K[m(H)/X] = b - c$. Now, $m(K[m(H)/X])$ has embedding relation $\{(u, *, *, q, d) \mid u \in \{b, c\},\ q \in A,\ d \in \{\text{in}, \text{out}\}\}$, but, according to Definition 3.1, $m(K)[m(H)/X]$ has embedding relation $\{(b, *, *, q, d) \mid q \in A,\ d \in \{\text{in}, \text{out}\}\}$, because there is no pair (X, q) in conn. Taking $M = a - X$ and $w = X$, we obtain in both sides of (i) above the graph $a - b - c$ with embedding $\{(a, *, *, q, d) \mid q \in A,\ d \in \{\text{in}, \text{out}\}\}$, but in (ii) we obtain $a - b - c$ as the left-hand side and $a - b - c$ with an additional edge between a and c as the right-hand side.

From these observations we can conclude that the good framework for investigating NLC-like grammars is that of NCE-substitutions (viz. $m(K)[m(H)/v]$) of graphs with embedding (as defined in Definition 3.1), rather than that of NLC-substitutions (viz. $K[m(H)/v]$) of graphs without embedding (as in [6]). The confluent NLC grammars are *actually context-free* (i.e., confluent and associative, see [6]) in the appropriate framework, i.e., that of NCE-substitution, and there is no reason to consider with special attention the context-free NLC grammars of [6] (i.e., the confluent NLC grammars that are also associative with respect to NLC-substitution).

We now wish to compare the class S-HH with the class CFHG of hypergraph languages generated by the context-free hypergraph grammars (or hyperedge replacement grammars) of [1, 26, 27, 28].

A translation from hypergraphs to graphs was defined in [25]. More precisely, in [25] a one-to-one mapping "gra" from hypergraphs to graphs is defined such that $L \in \text{CFHG}$ if and only if $\text{gra}(L) \in \text{B-edNCE}_{bntd}$ (where B-edNCE_{bntd} is a subclass of B-edNCE, and hence of C-edNCE). In this translation a vertex of a hypergraph H is represented by a vertex of $\text{gra}(H)$, and so is an edge. Appropriate labels distinguish the two kinds of vertices of $\text{gra}(H)$.

Let us now consider a monadic second-order formula ϕ written with quantification over edges and sets of edges, as well as vertices and sets of vertices. It can be translated into a formula ϕ' such that for every hypergraph H, $\text{gra}(H)$ satisfies ϕ' iff H

satisfies ϕ, where ϕ' uses quantification over vertices and sets of vertices only. This is possible because the vertices of gra(H) encode the vertices as well as the edges of H. It follows that Theorem 6.9(1, 2) entails the corresponding results for CFHG proved in [10] (see also [7–9]); to see this for (1), one has to see, as in the previous case, that the B-edNCE$_{bntd}$ property is preserved by the constructions.

Since CFHG and S-HH are both classes of hypergraph languages, it is also interesting to compare them directly. We have already seen that S-HH contains graph languages that are not in CFHG (such as the set of all complete graphs). In the other direction there is a trivial answer: the hypergraphs generated by context-free hypergraph grammars may have multiple hyperedges with the same label and they may have hyperedges with no vertices, whereas the ones we consider in the present paper may not (they are "simple"). However, we will show that even when we restrict ourselves to our (simple) hypergraphs, CFHG contains a hypergraph language that is not in S-HH. For this purpose it is convenient to define a CFHG grammar as a particular kind of HH grammar, in the following way. It is straightforward to show, by standard techniques, that, restricted to simple hypergraphs, this definition is equivalent to the usual ones. A HH grammar is *identification-free* if, for every production $X \to (H, \text{port})$, the sets port(i), $i \in \mathbb{N}_+$, are pairwise disjoint.

DEFINITION 7.3. A *context-free hypergraph grammar* (abbreviated CFHG grammar) is a loop-free, identification-free HH grammar $G = (N, A, P, X_{\text{in}})$ for which there exists a mapping rank: $N \to \mathbb{N}$ with $\text{rank}(X_{\text{in}}) = 0$, such that for every production $X \to (H, \text{port})$ in P:

(1) $\text{rank}_H(e) = \text{rank}(\text{lab}_H(e))$ for every nonterminal edge e of H, and

(2) $\text{card}(\text{port}(i)) = 1$ for every $i \in [1, \text{rank}(X)]$ and $\text{card}(\text{port}(i)) = 0$ for every $i > \text{rank}(X)$.

CFHG grammars are edge rewriting rather than handle rewriting: one may imagine that the nonterminal vertices of an edge e and their incident edges do not change during application of a production to e. For the assumption that a CFHG grammar is loop-free, see Theorem I.4.6 of [26]; see also Section 6 of [19]. For the assumption that it is identification-free, see Theorem 6 of [25] or Lemma 3.2 of [18]. Examples of CFHG grammars are G_1 of Fig. 6 (with $\text{rank}(X_2) = 2$) and G_4 of Fig. 12 (with $\text{rank}(X) = 2$).

In certain cases CFHG hypergraph languages are in S-HH.

THEOREM 7.4. *Let L be a hypergraph language, generated by a* CFHG *grammar G. If G is separated, or L is a graph language, then L is in* S-HH.

Proof. The case that G is separated is trivial. Now assume that L is a graph language. It is shown in [25] that L is in B-edNCE. Hence, since B-edNCE $\subseteq$ C-edNCE, Theorem 4.4 shows that $L \in$ S-HH. It would be interesting to have a direct construction in this case, i.e., one that does not make use of C-edNCE grammars. ∎

It can be shown, by a complicated proof, that all CFHG hypergraph languages of bounded degree are also in S-HH (where the degree of a vertex of a hypergraph is the number of edges incident with that vertex).

We shall now prove that there exist CFHG hypergraph languages that are not in S-HH. (Note that such languages are not generated by a separated CFHG grammar, are not graph languages, and are not of bounded degree.) Together with the fact that there exist graph languages in S-HH that are not in CFHG, this shows that the classes CFHG and S-HH are incomparable subclasses of the class of HH languages. Since both classes are context-free, in the sense of [6], it would be interesting to find a natural context-free class containing them both.

THEOREM 7.5. *There exists a hypergraph language that is in* CFHG *but not in* S-HH.

The remainder of this section consists of the proof of Theorem 7.5. The hypergraph language in Theorem 7.5 will be a variation of the graph language generated by the example CFHG grammar G_4 (in Fig. 12).

For a hypergraph H we define the hypergraph plus(H) by turning each edge of rank k into an edge of rank $k+1$, with an additional vertex. For technical reasons we add edges of rank 1, with label \$, to the new vertices. If $H=(V, E)$ is a hypergraph in HG(A) and \$ is a "new" symbol, then plus(H) $=(V', E')$ is a hypergraph in HG($A \cup \{\$\}$), where $V' = V \cup E$, and $E' = \{(a, v_1, ..., v_k, e) \mid e \in E, e=(a, v_1, ..., v_k)\} \cup \{(\$, e) \mid e \in E\}$.

In the next lemma, S-CFHG denotes the class of languages generated by separated CFHG grammars.

LEMMA 7.6. (1) *For every hypergraph language* L, plus(L) $\in$ CFHG *if and only if* $L \in$ CFHG. *The same holds for* S-CFHG.

(2) *For every hypergraph language* L, plus(L) $\in$ S-HH *if and only if* $L \in$ S-CFHG.

(3) *The graph language* L(G_4) *is in* CFHG *but not in* S-CFHG.

This lemma implies Theorem 7.5, as follows: plus(L(G_4)) is in CFHG by (3) and (1); if it would be in S-HH then L(G_4) $\in$ S-CFHG by (2), which contradicts (3).

To prove the lemma we need the following technical result. We say that a CFHG grammar $G=(N, A, P, X_{in})$ has a *port parasite* if there exist a production $X \to (H, \text{port})$ of G and a terminal edge e of H such that all vertices of e are ports (i.e., $\text{vset}_H(e) \subseteq \bigcup \{\text{port}(i) \mid i \in \mathbb{N}_+\}$). G_1 has port parasites, but G_4 has not.

LEMMA 7.7. *For every (separated)* CFHG *grammar there exists an equivalent (separated)* CFHG *grammar that has no port parasites.*

Proof (sketch). Let $G=(N, A, P, X_{in})$ be a CFHG grammar. Consider a sentential form K of G, a nonterminal edge e of K, and a derivation $K \overset{*}{\Rightarrow} H$, where

H is terminal. The derivation steps that concern e and its descendants derive a set T of terminal edges of H between the nonterminal vertices incident with e. The edges in T are necessarily produced by productions with port parasites. A new grammar G' can be constructed in which the edges of T are produced as soon as the vertices of e are produced (and hence are already present in K). G' computes the set T, for each nonterminal edge, from right to left in the derivation (i.e., in a "bottom-up" fashion). The formal construction is straightforward. ▮

Proof of Lemma 7.6(1). Let L be a hypergraph language. It is easy to see that if $L \in \mathrm{CFHG}$ then $\mathrm{plus}(L) \in \mathrm{CFHG}$: just turn all terminal edges of rank k into edges of rank $k+1$ in the right-hand sides of the productions. In fact, the easiest way to do this is to replace each production $X \to (H, \mathrm{port})$ by $X \to (\mathrm{plus}(H), \mathrm{port})$.

Now let $G = (N, A \cup \{\$\}, P, X_{\mathrm{in}})$ be a CFHG grammar with $\mathrm{L}(G) = \mathrm{plus}(L)$. By Lemma 7.7 we may assume that G has no port parasites. In particular, no port is incident with a \$-labeled edge of rank 1. Thus, if a nonterminal vertex, in a sentential form K, is not incident with a \$-labeled edge, then it will not be incident with such an edge in the terminal hypergraph generated from K.

We first construct an equivalent CFHG grammar G'' such that in each nonterminal the information is stored which of its nonterminal vertices are incident with a \$-labeled edge: $G'' = (N'', A \cup \{\$\}, P'', X''_{\mathrm{in}})$, where $N'' = \{(X, S) \mid S \subseteq [1, \mathrm{rank}(X)]\}$, $X''_{\mathrm{in}} = (X_{\mathrm{in}}, \varnothing)$, and if $X \to (H, \mathrm{port})$ is in P, then, for every $S \subseteq [1, \mathrm{rank}(X)]$, $(X, S) \to (H'', \mathrm{port})$ is in P'', where H'' is obtained from H by relabeling every nonterminal edge e by (X_e, S_e), where $X_e = \mathrm{lab}_H(e)$ and $S_e = \{i \in [1, \mathrm{rank}(X_e)] \mid \mathrm{vert}_H(e, i)$ is incident with a \$-labeled edge in H or $\mathrm{vert}_H(e, i)$ is a j-port with $j \in S_e\}$.

To obtain a CFHG grammar G' with $\mathrm{L}(G') = L$ we transform G'' by turning every terminal edge of rank $k+1$ into one of rank k, and removing all vertices that are incident with a \$-labeled edge. Formally, we define $G' = (N'', A, P', X''_{\mathrm{in}})$ such that, in G', $\mathrm{rank}(X, S) = \mathrm{card}([1, \mathrm{rank}(X)] - S)$, and P' is constructed as follows. If $(X_0, S_0) \to (V, E, \mathrm{port})$ is a production in P'', then $(X_0, S_0) \to (V', E', \mathrm{port}')$ is in P', where

$$V' = \{v \in V \mid v \text{ is not incident with a \$-labeled edge and there is no } i \in S_0 \text{ such that } (i, v) \in \mathrm{port}\},$$

$$E' = \{(a, v_1, ..., v_k) \mid a \in A,\ (a, v_1, ..., v_k, v) \in E \text{ for some } v \in V\} \cup \{((X, S), v_{i_1}, ..., v_{i_m}) \mid \exists e \in E:\ \mathrm{lab}(e) = (X, S),\ [1, \mathrm{rank}(X)] - S = \{i_1, ..., i_m\},\ i_1 < i_2 < \cdots < i_m, \text{ and } \mathrm{vert}(e, i_j) = v_{i_j} \text{ for } j \in [1, m]\},$$

$$\mathrm{port}' = \{(j, v) \mid j \in [1, m], (i_j, v) \in \mathrm{port}\}, \text{ where } [1, \mathrm{rank}(X_0)] - S_0 = \{i_1, ..., i_m\}, \text{ with } i_1 < \cdots < i_m.$$

This ends the construction of G'. ▮

Proof of Lemma 7.6(2). If $L \in \text{S-CFHG}$ then $\mathrm{plus}(L) \in \text{S-CFHG}$ by Lemma 7.6(1), and hence $\mathrm{plus}(L) \in \text{S-HH}$. The proof of the other direction is based on a result from [20]. A hypergraph language L is of *bounded hyper-degree* if there is an

integer b such that for every $k \geqslant 2$, for every $j \in [1, k]$, for every $H \in L$, and for all vertices $v_1, ..., v_{j-1}, v_{j+1}, ..., v_k$ of H: $\text{card}(\{e \in E_H \mid \text{rank}_H(e) = k, \text{vert}_H(e, i) = v_i \text{ for all } i \in \{1, ..., j-1, j+1, ..., k\}\}) \leqslant b$. It is easy to see that, for every hypergraph language L, plus (L) is of bounded hyper-degree. In fact, for $j < k$ the bound is 1, because no two distinct vertices e can have the same vert$(e, \text{rank}(e))$, and for $j = k$ the bound is card(A), where $L \subseteq \text{HG}(A)$, because in a hypergraph of L there are at most card(A) edges with the same vertices.

Suppose that plus$(L) \in$ S-HH. It is shown in [20] that every hypergraph language of bounded hyper-degree that is in S-HH, is in S-CFHG. Hence plus(L) is in S-CFHG, and so, by Lemma 7.6(1), $L \in$ S-CFHG. ∎

To show Lemma 7.6(3) we will use the following result. We say that a B-edNCE grammar is 3-*separated* if in each sentential form the (undirected) distance between two distinct nonterminal vertices is $\geqslant 3$ (see [23]).

LEMMA 7.8. *For every graph language L, if $L \in$ S-CFHG, then L is generated by a 3-separated* B-edNCE *grammar.*

Proof. In [25] an indirect construction is given showing that every CFHG graph language is in B-edNCE. It is difficult to see whether that construction preserves (3-)separatedness. For that reason we here give a direct construction that does.

Let $G = (N, A, P, X_{\text{in}})$ be an S-CFHG grammar generating L. By Lemma 7.7 we may assume that G has no port parasites. Since L(G) is a graph language, this means that every vertex in the right-hand side of a production that is not a port is incident with an edge of rank 1 (the label of the vertex). An equivalent edNCE grammar $G' = (N', A, P', X_{\text{in}})$ can easily be constructed by turning the nonterminal edges of G into nonterminal vertices of G' (with their tentacles turned into edges) and by turning all vertices of G into terminal vertices of G'. N' is defined by $N' = N \cup [1, m]$, where m is the maximal rank of a nonterminal edge of H; the elements of $[1, m]$ will be used as edge labels only. P' is constructed as follows:

Let $X_0 \to (H, \text{port})$ be in P. Since L(G) is a graph language, we may assume that all terminal edges of H are of the form $(a, v_1, ..., v_k)$ with $k = 1$ or $k = 2$. Then P' contains the production $X_0 \to (V, E, \text{emb})$ such that $V = (V_H - R) \cup E_N$, where R is the set of ports of H and E_N is the set of nonterminal edges of H, and

$$\begin{aligned} E = {} & (E_H - E_N) \cup \{(X, e) \mid e \in E_N, \text{lab}_H(e) = X\} \\ & \cup \{(i, e, v) \mid e \in E_N, v \in V_H - R, \text{vert}_H(e, i) = v\} \\ & \cup \{(i, v, e) \mid e \in E_N, v \in V_H - R, \text{vert}_H(e, i) = v\}. \end{aligned}$$

The embedding is defined as follows: Let $p_i \in R$ be the unique i-port of H, for every $i \in [1, \text{rank}(X_0)]$. Then

$$\begin{aligned} \text{emb} = {} & \{(v, i, a, q, \text{out}) \mid v \in V_H - R, (a, v, p_i) \in E_H, \quad a, q \in A\} \\ & \cup \{(v, i, a, q, \text{in}) \mid v \in V_H - R, (a, p_i, v) \in E_H, \quad a, q \in A\} \\ & \cup \{(e, i, j, q, d) \mid e \in E_N, \text{vert}_H(e, j) = p_i, q \in A, d \in \{\text{in}, \text{out}\}\}. \end{aligned}$$

This ends the definition of G'. Edges (and tentacles) that are incident with non-ports only, are produced by the right-hand side of the production of G', whereas the other edges (and tentacles) are produced by the embedding.

For a sentential form H of G, define the graph $g(H) \in \mathrm{GR}(N' \cup A)$ to be (V, E), where V and E are as above in the definition of P', but with $R = \varnothing$. It can be shown that $\mathrm{se}(X_{\mathrm{in}}) \overset{*}{\Rightarrow} K'$ in G' iff $\exists K$: $\mathrm{se}(X_{\mathrm{in}}) \overset{*}{\Rightarrow} K$ in G and $g(K) = K'$ (where, in G', the derivations should be restricted appropriately, cf. Remark 2.3). This shows that $\mathrm{L}(G') = \mathrm{L}(G)$. It also shows that G' is 3-separated. In fact, since G is separated, every sentential form K of G is separated; and, hence, two distinct nonterminal vertices of a sentential form $g(K)$ of G' have distance $\geqslant 3$. ∎

Proof of Lemma 7.6(3). It is shown in [23] that $\mathrm{L}(G_4)$ cannot be generated by a 3-separated B-edNCE grammar. Hence, by Lemma 7.8, $\mathrm{L}(G_4)$ is not in S-CFHG. ∎

REFERENCES

1. M. BAUDERON AND B. COURCELLE, Graph expressions and graph rewritings, *Math. Systems Theory* **20** (1987), 83–127.
2. F. J. BRANDENBURG, On partially ordered graph grammars, in [14, pp. 99–111].
3. F. J. BRANDENBURG, On polynomial time graph grammars, *in* "Proceedings, STACS 88," Lect. Notes in Comput. Sci., Vol. 294, pp. 227–236, Springer-Verlag, New York/Berlin, 1988.
4. D. G. CORNEIL, H. LERCHS, AND L. S. BURLINGHAM, Complement reducible graphs, *Discrete Appl. Math.* **3** (1981), 163–174.
5. B. COURCELLE, Equivalences and transformations of regular systems—Applications to recursive program schemes and grammars, *Theoret. Comput. Sci.* **42** (1986), 1–122.
6. B. COURCELLE, An axiomatic definition of context-free rewriting and its application to NLC graph grammars, *Theoret. Comput. Sci.* **55** (1987), 141–181.
7. B. COURCELLE, The monadic second-order logic of graphs. III. Tree-decompositions, minors, and complexity issues, *RAIRO Inform. Théor. Appl.* **26** (1992), 257–286.
8. B. COURCELLE, Graph rewriting: An algebraic and logical approach, *in* "Handbook of Theoretical Computer Science, Vol. B" (J. van Leeuwen, Ed.), pp. 193–242, Elsevier, Amsterdam, 1990.
9. B. COURCELLE, The monadic second-order logic of graphs: Definable sets of finite graphs, Lect. Notes in Comput. Sci., Vol. 344, pp. 30–53, Springer-Verlag, New York/Berlin, 1989.
10. B. COURCELLE, The monadic second-order logic of graphs. I. Recognizable sets of finite graphs, *Inform. and Comput.* **85** (1990), 12–75; see also [14, pp. 112–133].
11. B. COURCELLE, Graphs as relational structures: An algebraic and logical approach, in [13, pp. 238–252].
12. B. COURCELLE, J. ENGELFRIET, AND G. ROZENBERG, Context-free handle-rewriting hypergraph grammars, in [13, pp. 253–268].
13. H. EHRIG, H.-J. KREOWSKI, AND G. ROZENBERG (Eds.), "Graph-Grammars and Their Application to Computer Science," Lect. Notes in Comput. Sci., Vol. 532, Springer-Verlag, New York/Berlin, 1991.
14. H. EHRIG, M. NAGL, G. ROZENBERG, AND A. ROSENFELD (Eds.), "Graph-Grammars and Their Application to Computer Science," Lect. Notes in Comput. Sci., Vol. 291, Springer-Verlag, New York/Berlin, 1987.
15. H. EHRIG AND G. ROZENBERG, Some definitional suggestions for parallel graph grammars, *in* "Automata, Languages, and Development" (A. Lindenmayer, G. Rozenberg, Eds.), pp. 443–468, North-Holland, Amsterdam, 1976.

16. J. ENGELFRIET, Context-free NCE graph grammars, *in* "Proceedings, FCT '89," Lect. Notes in Comput. Sci., Vol. 380, pp. 148–161, Springer-Verlag, New York/Berlin, 1989.
17. J. ENGELFRIET, A characterization of context-free NCE graph languages by monadic second-order logic on trees, in [13, pp. 311–327].
18. J. ENGELFRIET AND L. M. HEYKER, The string generating power of context-free hypergraph grammars, *J. Comput. System Sci.* **43** (1991), 328–360.
19. J. ENGELFRIET AND L. M. HEYKER, Context-free hypergraph grammars have the same term-generating power as attribute grammars, *Acta Informatica* **29** (1992), 161–210.
20. J. ENGELFRIET AND L. M. HEYKER, "Hypergraph Languages of Bounded Degree," Report 91-01, Leiden, 1991, *J. Comput. System Sci.*, to appear.
21. J. ENGELFRIET AND G. LEIH, Linear graph grammars: Power and Complexity, *Inform. and Comput.* **81** (1989), 88–121.
22. J. ENGELFRIET, G. LEIH, AND G. ROZENBERG, Apex graph grammars, in [14, pp. 167–185].
23. J. ENGELFRIET, G. LEIH, AND G. ROZENBERG, Nonterminal separation in graph grammars, *Theoret. Comput. Sci.* **82** (1991), 95–111.
24. J. ENGELFRIET, G. LEIH, AND E. WELZL, Boundary graph grammars with dynamic edge relabeling, *J. Comput. System Sci.* **40** (1990), 307–345.
25. J. ENGELFRIET AND G. ROZENBERG, A comparison of boundary graph grammars and context-free hypergraph grammars, *Inform. and Comput.* **84** (1990), 163–206.
26. A. HABEL, "Hyperedge Replacement: Grammars and Languages," Ph.D. thesis, Bremen, 1989.
27. A. HABEL AND H.-J. KREOWSKI, Some structural aspects of hypergraph languages generated by hyperedge replacement, *in* "Proceedings, STACS '87," Lect. Notes in Comput. Sci., Vol. 247, pp. 207–219, Springer-Verlag, New York/Berlin, 1987.
28. A. HABEL AND H.-J. KREOWSKI, May we introduce to you: Hyperedge replacement, in [14, pp. 15–26].
29. D. JANSSENS, H.-J. KREOWSKI, G. ROZENBERG, AND H. EHRIG, "Concurrency of Node-Label-Controlled Graph Transformations," Report 82–38, University of Antwerp, 1982.
30. D. JANSSENS AND G. ROZENBERG, On the structure of node-label-controlled graph languages, *Inform. Sci.* **20** (1980), 191–216.
31. D. JANSSENS AND G. ROZENBERG, A survey of NLC grammars, *in* "Proceedings, CAAP '83," Lect. Notes in Comput. Sci., Vol. 159, pp. 114–128, Springer-Verlag, New York/Berlin, 1983.
32. M. KAUL, "Syntaxanalyse von Graphen bei Präzedenz-Graph-Grammatiken," Ph.D. thesis, Osnabrück, 1985.
33. C. LAUTEMANN, Efficient algorithms on context-free graph languages, *in* "Proceedings, ICALP '88," Lect. Notes in Comput. Sci., Vol. 317, pp. 362–378, Springer-Verlag, New York/Berlin, 1988.
34. M. G. MAIN AND G. ROZENBERG, Handle NLC grammars and r.e. languages, *J. Comput. System. Sci.* **35** (1987), 192–205.
35. M. G. MAIN AND G. ROZENBERG, Edge-label controlled graph grammars, *J. Comput. System Sci.* **40** (1990), 188–228. See also [14, pp. 411–426].
36. J. MEZEI AND J. B. WRIGHT, Algebraic automata and context-free sets, *Inform. and Control* **11** (1967), 3-29.
37. U. MONTANARI AND F. ROSSI, An efficient algorithm for the solution of hierarchical networks of constraints, in [14, pp. 440–457].
38. M. NAGL, "Graphgrammatiken," Vieweg, Braunschweig, 1979.
39. G. ROZENBERG AND E. WELZL, Boundary NLC graph grammars—Basic definitions, normal forms, and complexity, *Inform. and Control.* **69** (1986), 136–167.
40. R. SCHUSTER, "Graphgrammatiken und Grapheinbettungen: Algorithmen und Komplexität," Ph.D. thesis, Report MIP–8711, Passau, 1987.

Printed in Belgium
Uitgever: Academic Press, Inc.
Verantwoordelijke uitgever voor België:
Hubert Van Maele
Altenastraat 20, B-8310 Sint-Kruis